Markt&Technik

Jürgen Hückstädt

# CP/M 2.2 ANWENDER HANDBUCH

## CPC 464/664/6128

Ein unentbehrliches Nachschlagewerk für die praktische Arbeit mit dem Betriebssystem CP/M 2.2. Mit vielen CPC-systemspezifischen Informationen: Speicheraufteilung

- Schnittstellen
- Sprungtabellen

Jürgen Hückstädt

# CP/M 2.2 Anwender-Handbuch CPC 464/664/6128

Ein unentbehrliches Nachschlagewerk für die praktische Arbeit mit dem Betriebssystem CP/M 2.2. Mit vielen CPC-systemspezifischen Informationen:
Speicheraufteilung
Schnittstellen
Sprungtabellen

Markt & Technik Verlag

CIP-Kurztitelaufnahme der Deutschen Bibliothek

**Hückstädt, Jürgen:**
[CP-M-zwei-zwei-Anwenderhandbuch] CP-M-2.2-Anwenderhandbuch : CPC 464/664/6128 ;
e. unentbehrl. Nachschlagewerk für d. prakt. Arbeit mit d. Betriebssystem CP/M 2.2 ;
mit vielen CPC-systemspezif. Informationen: Speicheraufteilung, Schnittstellen, Sprungtab. / Jürgen Hückstädt. -
Haar bei München : Markt-und-Technik-Verlag, 1986.
ISBN 3-89090-204-9

Die Informationen im vorliegenden Buch werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen.
Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Verlag, Herausgeber und Autoren können
für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine
Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien.
Die gewerbliche Nutzung der in diesem Buch gezeigten Modelle und Arbeiten ist nicht zulässig.

CP/M' ist ein Warenzeichen der Digital Research Inc., USA

15 14 13 12 11 10 9 8 7 6 5 4 3 2
89 88 87

ISBN 3-89090-204-9

(©) 1986 by Markt & Technik, 8013 Haar bei München
Alle Rechte vorbehalten
Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Jantsch, Günzburg
Printed in Germany



# Inhaltsverzeichnis

Vorwort 9

1 Einführung
1.1 Was ist CP/M?
1.2 Geschichtliche Entwicklung
1.3 Gerätezusammenstellung
1.3.1 Bildschirm und Tastatur (Konsole)
1.3.2 Diskettenlaufwerk
1.3.3 Drucker
1.4 Wichtige Grundbegriffe für Anfänger
1.4.1 Der CP/M-Start
1.4.2 Die drei Grundelemente von CP/M

2 Allgemeine Grundlagen
2.1 Einige Tippversuche

3 STAT 55
3.1 Freie Diskettenkapazität 55
3.2 Zustand von Dateien 57
3.3 Laufwerkseigenschaften 58
3.4 Gerätezuordnung 60
3.5 Schreibschutz für Laufwerk 62
3.6 Dateiattribute 63
3.7 Benutzerbereiche ermitteln 64
3.8 Zusammenfassung der STAT-Befehle 64

| | | |
|---|---|---|
| 4 | Der Editor | 67 |
| 4.1 | Was ist ein Editor? | 67 |
| 4.2 | Das Dienstprogramm ED | 69 |
| 4.2.1 | Textdatei anlegen | 70 |
| 4.2.2 | Zeilen einfügen/löschen | 73 |
| 4.2.3 | Text korrigieren | 77 |
| 5 | PIP - ein universelles Kopierprogramm | 79 |
| 5.1 | Allgemeines | 79 |
| 5.2 | Laden und Aktivieren von PIP | 80 |
| 5.3 | Kopieren von Diskette zu Diskette | 81 |
| 5.4 | Dateien aneinanderhängen | 83 |
| 5.5 | Datenaustausch mit anderen Peripheriegeräten | 85 |
| 5.6 | Spezielle Befehle | 88 |
| 5.6.1 | Dateiausschnitte | 88 |
| 5.6.2 | Groß- und Kleinschrift | 89 |
| 5.6.3 | Textausgabe mit Zeilennummern | 90 |
| 5.6.4 | Verify | 91 |
| 5.6.5 | Kopieren geschützter Dateien | 92 |
| 5.6.6 | Zuammenfassen binärer Dateien | 92 |
| 6 | Stapelverarbeitung | 93 |
| 6.1 | Allgemeines | 93 |
| 6.2 | SUBMIT | 93 |
| 6.3 | XSUB | 98 |
| 7 | Debugging | 101 |
| 7.1 | DUMP | 102 |
| 7.2 | DDT | 104 |
| 7.2.1 | DDT laden | 104 |
| 7.2.2 | SAVE | 106 |
| 7.2.3 | Speicher ansehen | 109 |
| 7.2.4 | Speicher ändern | 110 |
| 7.2.5 | Assembler/Disassembler | 111 |
| 7.2.6 | Weitere Möglichkeiten | 112 |
| 8 | Assemblerprogrammierung | 115 |
| 8.1 | Ein paar Worte vorweg | 115 |
| 8.2 | Der Quelltext | 116 |
| 8.3 | Quelltext assemblieren | 120 |





| | | |
|---|---|---|
| 9 | Hinter den Kulissen | 125 |
| 9.1 | Interne Speicherorganisation | 125 |
| 9.2 | Directory und File-Control-Block (FCB) | 127 |
| 9.3 | BDOS-Aufrufe | 129 |
| 9.4 | Mehr über das BIOS | 134 |
| 9.5 | Anpassen von CP/M-Software | 138 |
| 10 | CP/M-Befehlsübersicht | 139 |
| | AMSDOS | 140 |
| | ASM | 141 |
| | BOOTGEN | 143 |
| | CHKDISC | 144 |
| | CLOAD | 145 |
| | COPYDISC | 146 |
| | CSAVE | 147 |
| | DDT | 148 |
| | DIR | 157 |
| | DISCCHK | 158 |
| | DISCCOPY | 159 |
| | DISCKIT2 | 160 |
| | DUMP | 161 |
| | ED | 162 |
| | ERA | 172 |
| | FILECOPY | 173 |
| | FORMAT | 174 |
| | LOAD | 175 |
| | MOVCPM | 176 |
| | PIP | 177 |
| | REN | 187 |
| | SAVE | **188** |
| | SETUP | 190 |
| | STAT | 193 |
| | SUBMIT | 200 |
| | SYSGEN | 202 |
| | TYPE | 203 |
| | USER | 204 |
| | XSUB | 205 |

Anhang 207
Steuerzeichen für CP/M-Befehlszeile 207
ASCII-Code-Tabelle 208
CTRL-/ASCII-Code 209

Stichwortverzeichnis 211

Übersicht weiterer Markt&Technik-Bücher 213



## Vorwort

Die Schneider CPC-Computer sind universell einsatzbere Rechner, die neben der Standard-Programmiersprache BASIC auch unter CP/M 2.2 arbeiten können. Da es sich hierbei um ein weitverbreitetes Betriebssystem handelt, gibt es eine große Anzahl von CP/M-Programmen, die Sie nun auch auf Ihrem CPC ausführen können.

Dieses Buch ist in erster Linie für alle CP/M-Anwender geschrieben, die einen CPC 464 oder 664 besitzen, denn für diese Computer ist das CP/M 2. 2-Betriebssystem im Lieferumfang mit enthalten. Falls Sie jedoch mit dem CPC 6128 arbeiten, dürfte für Sie CP/M Plus von vorrangiger Bedeutung sein. Es gibt allerdings CP/M-Software, die nicht unter CP/M Plus lauffähig ist, weshalb Sie auch beim CPC 6128 auf die Version 2.2 zurückgreifen müssen, die ebenfalls zusammen mit dem CPC 6128 geliefert wird. In diesem Fall haben sämtliche hier gezeigten Verfahren und Anwendungen auch uneingeschränkt für den CPC 6128 Gültigkeit.

Wenn Sie mit diesem Buch arbeiten, benötigen Sie keinerlei Grundkenntnisse über CP/M. Es ist allerdings vorteilhaft, wenn Sie sich etwas mit BASIC auskennen. Obwohl BASIC und CP/M grundsätzlich verschieden sind, wollen wir in diesem Buch den Versuch wagen, Ihnen den Übergang zu CP/M so leicht wie möglich zu machen.

Die ersten beiden Kapitel vermitteln wichtige Grundbegriffe, die für das Arbeiten mit CP/M unerläßlich sind. Wenn Sie diese Kapitel gründlich durchgearbeitet haben, sind Sie bereits in der Lage, fertig gekaufte und für den CPC angepaßte CP/M-Software einzusetzen. Vielleicht möchten Sie mit WORDSTAR Texte erfassen, mit dBASEII eine Datenbank auf-bauen oder mit MULTIPLAN Ihr Haushaltsgeld verwalten. Diese weitverbreitete CP/M-Standard-Software ist heute für ca. 200 bis 300 DM für den CPC fertig angepaßt erhältlich (Verlag MARKT&TECHNIK), während sie bis vor kurzem noch weit über 1000 DM kostete!

Spätestens dann jedoch, wenn Sie mit diesen Programmen arbeiten, wer-den Sie tiefer in das CP/M-Betriebssystem einsteigen und die letzten Möglickeiten aus ihm herausholen wollen. CP/M bietet nämlich eine

Vielzahl von Utilities, die das Arbeiten erleichtern und mit denen wir uns in den Kapiteln 3 bis 8 eingehend beschäftigen werden.

Das 9. Kapitel ist für all diejenigen bestimmt, denen die Standardanwendungen noch nicht genügen und die sich für den internen Aufbau von CP/M interessieren. Die hier vermittelten Kenntnisse sind besonders für Maschinenprogrammierer von Bedeutung, die sich eigene CP/M-Programme schreiben möchten.

Das Buch schließt mit einer ausführlichen CP/M-Befehlsübersicht mit zahlreichen Beispielen. Besonders dem Profi steht somit noch zusätzlich ein Nachschlagewerk zur Verfügung, aus dem er sich alle notwendigen Informationen schnell beschaffen kann.

Ganz gleich, ob Sie bereits ein erfahrener CP/M-Anwender sind oder nicht, Sie erfahren in diesem Buch alles, was Sie zum Arbeiten mit CP/M auf dem Schneider CPC benötigen. Sie werden bald bemerken, daß CP/M kein abstrakter Begriff mehr für Sie ist, sondern das Tor zu einer Reihe neuer Anwendungsmöglichkeiten öffnet, die Ihnen viel Freude bereiten werden.

Der Autor



# 1 Einführung

Lieber Leser, Sie haben sich einen Schneider CPC 464 oder 664 zugelegt und zusätzlich dieses Buch gekauft, weil Sie sich näher mit CP/M beschäftigen möchten. Vielleicht gehören Sie zu den Anwendern, die bereits früher mit einem anderen CP/M-Computer gearbeitet haben, was Sie jetzt auch auf dem CPC tun möchten. Sie wollen in erster Linie Ihre Kenntnisse in CP/M vertiefen und interessieren sich für die systemspezifischen Eigenschaften des Schneider CPC unter CP/M.

Wahrscheinlich gehören Sie aber zu dem weitaus größeren Anwenderkreis, der sich den CPC zulegte, um darauf vorwiegend in BASIC zu programmieren, das ja mit seinen vielen Graphik- und Soundbefehlen in der Tat sehr leistungsstark ist. Als Sie sich für den Kauf des CPC entschlossen hatten, erfuhren Sie vielleicht nebenbei, daß dieser Computer auch CP/M-fähig ist. Sie konnten sich jedoch unter CP/M nichts konkretes vorstellen und schrieben zunächst, wie geplant, BASIC-Programme oder setzten fertig gekaufte Software ein. Im Laufe der Zeit hörten Sie jedoch immer mehr über CP/M, bis Sie sich schließlich dazu entschlossen, sich näher damit zu befassen.

Ganz gleich, ob Sie nun zu der einen oder der anderen Gruppe gehören, Sie werden in diesem Buch alles finden, was Sie zum Einsatz von CP/M auf dem Schneider-CPC benötigen.

Bevor wir jedoch zu den Einzelheiten kommen, beachten Sie bitte folgen-den wichtigen Hinweis: Dieses Buch behandelt lediglich die CP/M-Version 2.2 für den CPC 464 und 664. Neuerdings gibt es auch den CPC 6128, der für das Arbeiten mit CP/M Plus ausgelegt ist. Dies ist eine CP/M-Version, die mehr als 64K Speicher benötigt und einen teilweise anderen Befehlssatz enthält. Über CP/M Plus auf dem CPC 6128 und Joyce ist ein ähnliches Buch im Verlag MARKT&TECHNIK erschienen. Durch die getrennte Behandlung von CP/M 2.2 und CP/M Plus erhält der Anwender einen wesentlich besseren Überblick über die Version, mit der er arbeitet.

## 1.1 Was ist CP/M?

Falls Sie zu der zweiten angesprochenen Anwendergruppe gehören, können Sie sich vermutlich unter dem Begriff CP/M noch nicht allzuviel vorstellen. Deshalb zunächst eine kurze Erläuterung dazu:

Wenn Sie mit BASIC arbeiten, erhalten Sie ein Programm, das Sie sofort ausführen können. Sie tippen nur die entsprechenden BASIC-Zeilen korrekt ein und nach dem Befehl RUN läuft das Programm ab.

Ganz so einfach geht es mit CP/M allerdings nicht, denn CP/M ist lediglich ein Betriebssystem. Um dies zu verstehen, wollen wir uns in groben Zügen mit dem Aufbau des Computersystems beschäftigen. Verlieren Sie aber nicht gleich den Mut! Die folgenden Erläuterungen sind so abgefaßt, daß sie für einen Laien bzw. Computerneuling leicht verständlich sind. Falls Sie sich bereits mit CP/M auskennen, können Sie die folgenden Absätze überspringen.

Da Sie vermutlich zumindestens Grundkenntnisse in der BASIC-Programmierung mitbringen, wollen wir nachfolgend versuchen, den Über-gang von BASIC zu CP/M zu vollziehen.

Wenn Sie Ihren Computer einschalten, erscheint eine Einschaltmeldung, das Wort "Ready" und ein gelbes Kästchen (Cursor) auf dem Bildschirm, d.h. Sie können sofort BASIC-Programme eingeben oder von Kassette bzw. Diskette laden und dann ausführen.

Es kommt aber nicht von ungefähr, daß dies so ist. Selbst wenn der Computer noch kein Programm enthält, arbeitet er dennoch unaufhörlich und führt ohne Unterbrechung interne Maschinenroutinen aus.

Der Computer setzt sich aus einer Vielzahl von Bausteinen zusammen. Der wichtigste ist die CPU Z80A, ein Mikroprozessor, der ständig arbeitet und interne Maschinenroutinen ausführt. Da er den Computer sozusagen "am Leben erhält", kann man ihn auch als sein "Herz" betrachten. Darüber hinaus wird die CPU noch von einigen anderen Bausteinen unterstützt, die besondere Aufgaben, meist für Steuerzwecke, übernehmen. Darunter fallen Bausteine, die den Datentransfer mit der Floppy oder Kassette ab-wickeln, den Drucker ansteuern, ein Videochip für die Bildschirmausgabe oder ein Soundchip zur Tonerzeugung, um nur einige Beispiele zu nennen.

Nicht zu vergessen sind die Speicherbausteine, die entweder fest vorprogrammierte Routinen enthalten (ROM) oder flüchtig sind (RAM). So sind in den ROMs sämtliche Routinen gespeichert, die der Computer benötigt, damit Sie direkt nach dem Einschalten BASIC-Programme eingeben und



abarbeiten können. Aus diesem Grund kann man aus einem ROM nur Daten lesen, nicht aber welche hineinschreiben. ROMs bezeichnet man deshalb auch als Nur-Lese-Speicher, aus denen man die fest vorprogrammierten Informationen lesen kann, die auch beim Abschalten des Computers nicht verloren gehen.

Beim RAM dagegen handelt es sich um einen Schreib-Lese-Speicher, d.h. man kann hier Daten hineinschreiben und auch wieder daraus lesen. Beim Ausschalten des Computers gehen sie allerdings verloren. Im RAM wer-den beispielsweise Ihre BASIC-Programme oder die Zeichen, die Sie jeweils auf dem Bildschirm sehen, abgelegt.

Sämtliche Routinen und Daten, mit denen der Computer arbeitet, sind in 8-Bit-Einheiten abgelegt. Ein Bit ist die kleinste Informationseinheit, die Sie sich auch als Schalter vorstellen können, der entweder ein- oder aus-geschaltet ist. Entsprechend nimmt ein Bit den Zustand 0 (aus) oder 1 (ein) an.

Jeweils acht Bit zusammengefaßt ergeben eine Speicherzelle im Computer, die man als Byte bezeichnet. Wenn nun Ihr Schneider-Computer 64K RAM enthält, so heißt dies, daß er über 64* 1024 = 65536 Bytes Schreib-Lesespeicher verfügt, die allerdings nicht alle für BASIC-Programme nutzbar sind. Der Buchstabe K steht hier für Kbyte, wobei dieses in der Computertechnik nicht 1000, sondern strenggenommen 1024 Bytes um-faßt.

Jedes Byte setzt sich aus 8 Bit zusammen, so daß es 2 hoch 8 oder 256 verschiedene Zustände annehmen kann. Um dies zu verdeutlichen, wollen wir hier ein paar Beispiele zur binären Zahlendarstellung betrachten:

0000 0000

ergibt den Wert Null, da sämtliche 8 Bits nicht gesetzt sind. Dagegen ergibt

1111 1111

den Wert 255, der sich folgendermaßen errechnet (von links nach rechts):

```
    1 * 128
  + 1 *  64
  + 1 *  32
  + 1 *  16
  + 1 *   8
  + 1 *   4
  + 1 *   2
  + 1 *   1
        255
```

Da hier sämtliche Bits gesetzt sind, ist 255 der größte Wert, der in einem Byte dargestellt werden kann. Darüber hinaus gibt es aber noch Zwischenwerte, wie im folgenden Beispiel, das die Zahl 114 binär darstellt:

0111 0010

oder

```
    0 * 128
  + 1 * 64
  + 1 * 32
  + 1 * 16
  + 0 * 8
  + 0 * 4
  + 1 * 2
  + 0 * 1

        114
```

Will man nun größere Zahlenwerte als 255 darstellen, so benötigt man zwei oder mehr Bytes, die zu einer Einheit zusammmengefaßt werden. Zwei Bytes können somit 256*256 = 65536 verschiedene Zustände an-nehmen, drei Bytes 256*256*256 = 16777216 Zustände usw. Von dieser Möglichkeit macht z.B. BASIC Gebrauch, um Fließkommazahlen intern abzulegen, wenn auch in einer etwas abgewandelten Form.

Mit den 256 verschiedenen Zuständen eines Bytes kann man aber noch andere Informationen außer Zahlen erfassen, was jedoch von Fall zu Fall vereinbart werden muß. Dies geschieht beispielsweise zur Darstellung von Buchstaben und Satzzeichen, wobei jedem Zeichen ein entsprechender binärer Wert zugeordnet ist (ASCII-Code). Auch Maschinenprogramme



setzen sich aus einer Vielzahl von Binärwerten zusammen, wobei jeder Wert einer bestimmten Operation entspricht, die jeweils zwischen eins und vier Bytes benötigt. Die restlichen Bytes beinhalten dann z.B. Werte, die für diese Operation erforderlich sind.

Sie sehen, daß man in den einzelnen Bytes im Speicher eine Vielzahl von Informationen ablegen kann, wobei natürlich in jedem Fall feststehen muß, ob es sich um eine Fließkommazahl, um Textzeichen oder um Maschinenprogramme handelt. Selbst ein BASIC-Programm setzt sich aus Bytewerten zwischen 0 und 255 zusammen.

Damit Sie nun direkt nach dem Einschalten BASIC-Programme eingeben und ausführen können, müssen folgende interne Routinen ausgeführt bzw. aktiviert werden:

- Betriebssystem
- Editor
- BASIC-Interpreter

Das Betriebssystem ist der eigentliche Kern des Computers und arbeitet auf unterer Ebene. Es fragt z.B. die Tastatur ab, welche Taste Sie gerade gedrückt haben, steuert den Datentransfer zu Drucker und Floppy, sendet Videosignale an den Bildschirm oder erzeugt Töne über den Lautsprecher. Dazu gehören selbstverständlich auch die bereits erwähnten Zusatzbausteine für bestimmte Aufgaben.

Mit dem Betriebssystem allein können Sie aber noch kein BASIC-Programm schreiben und ausführen, Sie können sich höchstens eigene Maschinenprogramme schreiben, die bestimmte Anweisungen an das Betriebssystem geben. Das ist sicher eine interessante Aufgabe für den Profi, der sich mit der direkten Programmierung der Z80-CPU oder den an-deren Bausteinen auskennt und umfangreiche Kenntnisse in der Assemblerprogrammierung mitbringen muß.

In den frühen Zeiten der Computertechnik bediente man sich solcher Programmiertechniken, die allerdings sehr aufwendig waren. Bald jedoch entwickelte man höhere Programmiersprachen, die das Programmieren erheblich vereinfachten. Zu diesen Programmiersprachen gehört auch das BASIC Ihres CPC-Computers.

Eine solche Programmiersprache bewirkt im Prinzip nichts anderes, als einfache Anweisungen - die, wie z.B. in BASIC, der üblichen algebraischen Schreibweise sehr nahe kommen - in eine Vielzahl von Maschinenbefehlen umzusetzen und diese durch das Betriebssystem ausführen zu lassen.

Kommen wir wieder auf den Schneider-Computer zurück: Zunächst muß einmal der BASIC-Programmtext so, wie Sie ihn eingeben, erfaßt und intern gespeichert werden. Dies ist die Aufgabe des Editors, der letztlich ebenfalls aus einem umfangreichen Maschinenprogramm besteht, mit dessen Hilfe Sie jede BASIC-Zeile eintippen und die Eingabe durch Drücken der ENTER-Taste beenden können. Dabei wird der BASIC-Text in einen speziellen Code umgewandelt, der zwar noch weit vom eigentlichen Maschinencode entfernt ist, aber als BASIC-Programm abgearbeitet werden kann.

Die Abarbeitung übernimmt dann der BASIC-Interpreter, ebenfalls ein umfangreiches Maschinenprogramm, welches diesen Code liest und die zugehörigen Maschinenroutinen des Betriebssystems aufruft.

Nur durch das Zusammenwirken von Editor, Interpreter und Betriebssystem können wir auf so einfache Weise mit BASIC arbeiten. Dabei brauchen wir uns über die Arbeitsweise dieser Routinen keinerlei Gedanken machen. Nebenbei sei bemerkt, daß es auch Compiler gibt, die den Programmtext direkt in ausführbaren Maschinencode übersetzen. Dies hat den Vorteil, daß die Programme dann wesentlich schneller ablaufen, da ein Interpreter nicht mehr benötigt wird. Nachteilig dagegen wirkt sich ein erhöhter Speicherplatzbedarf und die Tatsache aus, daß ein compiliertes BASIC-Programm nicht mehr gelistet und korrigiert werden kann.

Nachdem wir nun die Arbeitsweise des CPC-Computers unter BASIC in groben Zügen kennengelernt haben, fällt es nicht mehr schwer, auch CP/M zu verstehen. CP/M ist nämlich ein universelles platten- oder diskettenorientiertes Betriebssystem, das an fast jeden Computer angepaßt werden kann, der eine Z80 oder 8080 CPU besitzt. Programme, die auf einem CP/M-Computer erstellt wurden, sind, von geringen Anpassungen einmal abgesehen, auf jedem anderen CP/M-Computer ebenfalls lauf-fähig. Es ist daher nicht verwunderlich, daß es eine Unmenge von CP/MSoftware auf dem Markt gibt, die auf jedem CP/M-Computer lauffähig ist. Nicht umsonst wird behauptet, daß CP/M-Programme die größte Softwarebibliothek der Welt darstellen.

CP/M-Programme sind meist reine Maschinenprogramme, die auf dem CP/M-Betriebssysstem lauffähig sind und genormte Einsprungadressen verwenden. Um mit CP/M arbeiten zu können, ist es aber nicht unbedingt notwendig, daß Sie sich in der Maschinenprogrammierung auskennen, wenn Sie auf fertige Programme zurückgreifen. Darüber hinaus werden zu CP/M noch verschiedene Dienstprogramme mitgeliefert, die das Arbeiten erleichtern.



Unter CP/M können Sie in den verschiedensten Programmiersprachen programmieren, wenn Sie sich den jeweiligen Editor und Interpreter bzw. Compiler besorgen. So gibt es auch M-BASIC - eine BASIC-Version, die unter CP/M arbeitet, die allerdings nicht mit dem CPC-BASIC identisch ist - oder Programmiersprachen wie z.B. TURBO-PASCAL, FORTRAN und FORTH, um nur einige Beispiele zu nennen.

Die Renner unter CP/M sind allerdings professionelle Programme wie WORDSTAR, ein komfortables Textverarbeitungssystem, dBASEII, ein Datenbankprogramm und MULTIPLAN, ein Tabellenkalkulationsprogramm. Diese Programme sind, fertig angepaßt für CPC-Computer, bei MARKT&TECHNIK erhältlich.

Wenn Sie andere CP/M-Programme von anderen CP/M-Rechnern auf den CPC übertragen möchten, müssen Sie diese auf den CPC überspielen und eventuell anpassen. Dazu benötigen Sie jedoch umfangreichere Kenntnisse, weshalb wir auf die Vorgehensweise erst an späterer Stelle in diesem Buch eingehen werden. Zunächst jedoch ist es wichtig, daß Sie mit fertig angepaßter CP/M-Software und den mitgelieferten Dienstprogrammen arbeiten können.

## 1.2 Geschichtliche Entwicklung

Anfangs der siebziger Jahre steckte die Mikrocomputertechnik noch in den Kinderschuhen. Der 8080 Mikroprozessor, der Vorgänger des Z80, der sich in den Schneider-Computern befindet, war seinerzeit die neueste Errungenschaft. In der damaligen Zeit waren Computer noch relativ teuer und besaßen für unsere heutigen Begriffe nur einen kleinen RAM-Speicher.

Schon bald entstand der Wunsch, für alle Computer, die mit einem 8080 und später einem Z80 Prozessor ausgestattet waren, ein einheitliches Betriebssystem zu entwickeln, das sich mit wenig Mühe an die jeweilige Hardware (Computertyp) anpassen ließ.

Ein Angestellter der Firma Intel Corporation namens Gary Kildall entwickelte schließlich im Jahre 1974 die erste CP/M-Version, die aber gegenüber den heute verwendeten Versionen noch recht primitiv war. Er arbeitete seinerzeit an einer Dateiverwaltung, die er in der höheren Programmiersprache PL/M schrieb und verwendete CP/M lediglich zur Unterstützung.

Im Jahre 1975 kam die erste kommerziell angebotene CP/M-Version 1.4 auf den Markt, die zunächst aber wenig Beachtung fand. Darüber hinaus war sie nur auf den genormten 8-Zoll-IBM-Diskettenlaufwerken lauf-fähig und konnte nur schwer an andere Diskettenformate angepaßt wer-den. Die heute weit verbreiteten 5.25-Zoll- und 3-Zoll-Disketten gab es damals noch nicht.

Dieser Zustand änderte sich erst, als um 1979/80 die erweiterte CP/M-Version 2.2 herauskam und von der Firma Digital Research vertrieben wurde. Jetzt war CP/M ein universelles Betriebssystem, das auf die verschiedensten Diskettenformate angepaßt werden konnte. Dies hatte zur Folge, daß CP/M bald eine große Verbreitung fand. Somit konnten, von geringen Anpassungen einmal abgesehen, auf einem CP/M-Rechner entwickelte Programme leicht auf andere CP/M-fähige Computer übertragen werden. Es ist daher auch nicht verwunderlich, daß CP/M-Programme die größte Softwarebibliothek der Welt darstellen.

Obwohl auch heute noch die Version 2.2 als Standard gilt, wurde CP/M zwischenzeitlich weiterentwickelt. Für 8-Bit-Rechner gibt es außerdem die Version 3.0, die mehr als 64K RAM-Speicher benötigt.

Auch für 16-Bit-Rechner gibt es CP/M-Versionen, die aber nicht sehr weit verbreitet sind. Für die mit einem 8086 bzw. 8088 Prozessor ausgestatteten IBMPC-Computer und deren kompatiblen Geräte entstand CP/M 86, das sich allerdings nicht gegen MS-DOS durchsetzen konnte. Darüber hinaus sind auch Computer mit einem 68000 Prozessor (z.B. Apple-Macintosh) CP/M-fähig, wobei die Version CP/M-68K Anwendung findet.

Neben CP/M gibt es noch das Betriebssystem MP/M. MP/M ist CP/M sehr ähnlich und enthält noch einige Zusatzfunktionen. Während CP/M ein Einbenutzersystem ist, ist MP/M multitasking-fähig, d.h. es kann mehrere Aufgaben gleichzeitig erfüllen. MP/M wird überall dort eingesetzt, wo mehrere Anwender über verschiedene Konsolen (in der Regel Tastatur und Bildschirm) auf einen Computer zugreifen. Daneben hat ein einzelner Benutzer auch die Möglichkeit, mehrere Aufgaben gleichzeitig durchführen zu lassen, wie z.B. einen Text editieren und einen anderen auszudrucken.

Auch für MP/M gibt es eine 16-Bit-Version, die den Namen Concurrent CP/M 86 hat. Unter dieser Version können sogar MS-DOS-Programme eingesetzt werden.



## 1.3 Gerätezusammenstellung

Damit ein Computersystem unter CP/M arbeiten kann, muß es sich zumindest aus folgenden Geräteeinheiten zusammensetzen:

- o Konsole (Bildschirm und Tastatur)
- o Diskettenlaufwerk
- o Drucker

### 1.3.1 Bildschirm **und** Tastatur (Konsole)

Bildschirm und Tastatur werden bei CP/M meist zusammengefaßt und als Konsole bezeichnet, obwohl es sich eigentlich um zwei getrennte Geräte handelt. Die Tastatur kann sowohl der amerikanischen als auch der deutschen Norm angepaßt sein, was für die Funktionstüchtigkeit des Systems jedoch ohne Bedeutung ist.

Die Schneider-Computer werden in der Regel mit einer amerikanischen QWERTY-Tastatur geliefert, die keine deutschen Sonderzeichen enthält. Zur Bedienung des CP/M-Betriebssystems ist dies auch nicht unbedingt erforderlich. Wenn Sie jedoch mit einem Textverarbeitungsprogramm deutschsprachige Texte eingeben, ist eine deutsche DIN-Tastatur sehr zu empfehlen. Man kann zwar die deutschen Umlaute auch umschreiben, indem man z.B. für den Buchstaben "ä" "ae" setzt; ein solcher Text wäre aber sehr ungewohnt für uns, so daß Sie diese Methode nur im äußersten Notfall einsetzen sollten.

Das Textverarbeitungsprogramm WORDSTAR sieht in seiner auf die CPC-Computer angepaßten Version (Verlag MARKT&TECHNIK) die Möglichkeit vor, neben dem amerikanischen auch den deutschen Zeichensatz zu verwenden. Dabei wird nicht nur die Tastatur entsprechend angepaßt, sondern es werden gleichzeitig auch die deutschen Sonderzeichen auf dem Bildschirm erzeugt. Da hierbei einige Tasten ihre ursprüngliche Bedeutung verlieren, empfiehlt es sich, entsprechende Tastenaufkleber herzustellen.

### 1.3.2 Diskettenlaufwerk

Zum Arbeiten mit CP/M benötigen Sie mindestens ein Diskettenlaufwerk. Obwohl dies im Prinzip für die meisten Zwecke ausreicht, ist ein zweites Laufwerk sehr zu empfehlen, da sich damit wesentlich bequemer arbeiten

läßt. Der CPC 664 enthält bereits ein eingebautes Laufwerk; es besteht aber die Möglichkeit, über ein spezielles Adapterkabel ein weiteres Laufwerk vom Typ DDI-1 anzuschließen.

Der CPC 464 besitzt herstellungsmäßig nur einen Kassettenrekorder, mit dem CP/M aber nicht betrieben werden kann, da es sich hierbei ausdrücklich um ein platten- oder diskettenbezogenes Betriebssystem handelt. Aus diesem Grund müssen Sie sich für den CPC 464 die Diskettenlaufwerke separat besorgen. Das erste Laufwerk wird zusammen mit einem Floppy-Interface geliefert, das hinten in den Expansionsport eingesteckt wird. Das Interface ist jedoch für zwei Laufwerke ausgelegt und enthält bereits einen Anschluß für die Zuschaltung eines Zweitlaufwerkes, das Sie ebenfalls bei Ihrem Händler erwerben können.

Das 3-Zoll-Floppylaufwerk, das bei den Schneider-Computern Verwendung findet, gehört zu den kleinsten und kompaktesten Laufwerken, die es heute gibt. Dabei ist es genauso leistungsfähig wie seine größeren Brüder mit 8 und 5.25 Zoll.

Wenn Sie eine 3-Zoll-Diskette neben eine 5-Zoll-Diskette legen, fällt Ihnen nicht nur der Größenunterschied auf, sondern auch die Schutzhülle. Die eigentliche Diskette besteht nämlich aus einer empfindlichen Magnetscheibe, die gegen mechanische Verletzungen unbedingt geschützt werden muß!

Die Hülle der 5.25-Zoll-Disketten ist sehr biegsam; sie besitzt zwei Öffnungen für den Schreib-/Lesekopf und das Indexloch. Es verwundert da-her nicht, daß der Umgang mit solchen Disketten besonderer Sorgfalt bedarf. Auch sollten die Disketten nur zum Gebrauch aus ihrer Papierhülle, die die Öffnungen schützt, herausgenommen werden.

Die für die Schneider-Floppies verwendeten 3-Zoll-Disketten befinden sich in einem stabilen Plastikgehäuse, das bei weitem nicht so empfindlich ist wie die Hülle der 5.25-Zoll-Disketten. Zusätzlich werden die Öffnungen für den Schreib-/Lesekopf und das Indexloch mit einer Metallklappe verschlossen, die sich beim Einschieben der Diskette in das Laufwerk automatisch öffnet. An der Seite des Gehäuses befindet sich nämlich eine Führungsschiene mit einem weißen Schieber. Wenn Sie diesen Schieber ( vorsichtig!) betätigen, spannen Sie eine Feder und öffnen gleichzeitig die Klappe, wobei die Magnetscheibe sichtbar wird. Beim Loslassen des Schiebers schnappt die Klappe aufgrund der Federwirkung wieder zu.

Die 3-Zoll-Disketten sind beidseitig beschreibbar. Sie können aber immer nur eine Seite, nämlich Seite A oder Seite B, gleichzeitig nutzen. Das



Laufwerk greift jeweils auf die Seite zu, die beim Einschieben oben auf der Diskette sichtbar ist.

Das Einlegen der Diskette in das Laufwerk ist denkbar einfach. Sie schieben die Diskette so weit hinein, bis sie einrastet. Dies sollte aber nur dann geschehen, wenn sowohl der Computer, als auch die Floppy eingeschaltet sind, da es ansonsten unter ungünstigen Umständen zu Datenverlusten kommen kann.

Das Herausnehmen der Diskette ist genauso einfach: Sie drücken lediglich den Knopf rechts unter der Öffnung und die Diskette springt heraus.

Abgesehen von dem Schreibschutz für CP/M-Dateien, mit dem wir uns noch befassen werden, können Sie Ihre Disketten vor versehentlichem Beschreiben schützen, wenn Sie links oben auf jeder Seite das kleine, mit einer roten Klappe verschlossene Loch öffnen. Dies erreichen Sie, indem Sie mit einem spitzen Stift den seitlich angebrachten roten Schieber betätigen.

Ungeachtet des Schreibschutzes sollten Sie von allen wichtigen Programmen und Dateien mindestens eine Sicherheitskopie anfertigen, die sich nach Möglichkeit auf einer anderen Diskette befinden sollte. Trotz der hohen Aufzeichnungssicherheit ist es dennoch nicht völlig auszuschließen, daß Schreib- oder Lesefehler auftreten. Wahrscheinlicher als rein technische Fehler ist dabei "menschliches Versagen", wodurch unbeabsichtigt eine Diskette neu formatiert oder wichtige Dateien gelöscht oder über-schrieben werden. Auch in einem solchen Fall kann man dann auf die Sicherheitskopie zurückgreifen.

Im weiteren Verlauf dieses Buches werden wir uns mit dem Kopieren von Dateien und ganzen Disketten noch näher beschäftigen.

Einige Fremdfirmen bieten 5.25-Zoll-Laufwerke für Schneider-Computer an, die entweder nur als Zweitlaufwerk (Laufwerk B) eingesetzt werden können oder zusammen mit einem eigenen Betriebssystem geliefert wer-den und somit die Schneider-Laufwerke vollständig ersetzen.

Es stellt sich allerdings die Frage, welchen Nutzen diese Laufwerke bringen. Der Einsatz von 5.25-Zoll-Disketten alleine bringt keinen Vorteil, wie wir bereits gesehen haben. Solche Laufwerke sind nur dann sinnvoll, wenn Sie Daten mit einem anderen Computer austauschen möchten, dessen Floppy ein IBM-kompatibles Diskettenaufzeichnungsformat verwendet. Neben dem IBM PC gibt es noch eine Reihe anderer Computer, die mit diesem Format arbeiten oder es zumindest lesen können. Die

CPC-Floppy arbeitet normalerweise zwar nicht im IBM-Format, sie kann aber Disketten in diesem Format lesen und beschreiben.

In absehbarer Zeit werden sicher auch Plattenlaufwerke (Harddisc) für Schneider-Computer auf den Markt kommen, die anstelle der Disketten-laufwerke eingesetzt werden können. Im Gegensatz zur einer Diskette enthalten sie eine Festplatte, die meist nicht austauschbar ist. Harddiscs haben eine wesentlich größere Speicherkapazität als Disketten, die meist im Rahmen zwischen 10 und 20 Megabytes (Millionen Bytes) liegt. Dafür sind sie auch wesentlich teurer.

### 1.3.3 **Drucker**

Jedes CP/M-System sollte natürlich einen Drucker enthalten. Der NLQ-401 Drucker von Schneider ist ein leistungsfähiger und preisgünstiger Matrixdrucker, mit dem Sie auch Korrepondenzschrift (NLQ = Near Letter Quality) erzeugen können. Darüber hinaus kann er mit Hilfe von DIP-Schaltern auch auf den deutschen Zeichensatz umgeschaltet werden, was sich besonders für eine deutsche Textverarbeitung vorteilhaft auswirkt.

Matrixdrucker besitzen einzelne Nadeln, die für jedes Zeichen ein Punkt-raster auf das Papier drucken. In der NLQ-Betriebsart werden die einzelnen Zeichen nach einem bestimmten Schema zweimal überdruckt, wodurch ein sehr sauberes Schriftbild entsteht, das dem eines Typenraddruckers (s.u.) sehr nahe kommt.

Außer dem NLQ 401 können Sie aber jeden Drucker anschließen, der mit einer Centronics-Schnittstelle ausgerüstet ist. Dies ist bei den meisten handelsüblichen Druckern der Fall, die zudem auch den deutschen Zeichensatz enthalten.

Ein besonders schönes Druckbild erzeugen die Typenraddrucker, bei denen keine Punktraster wie bei Matrixdruckern, sondern Typen für die einzelnen Zeichen, die auf einem rotierenden Rad angeordnet sind, gedruckt werden. Ein Typenraddrucker arbeitet aber wesentlich langsamer als ein Matrixdrucker.

Als Weiteres gibt es noch Tintenstrahldrucker, die ähnlich wie Matrix-drucker arbeiten. Das Zeichenraster wird hier aber nicht von Nadeln, sondern von feinen Farbdüsen erzeugt, die kleine Farbtupfer auf das Papier spritzen.



Die neueste Errungenschaft sind die Laserdrucker, die die Druckfarbe mit einem Laserstrahl auf das Papier übertragen und dort einbrennen. Diese Drucker arbeiten zwar extrem schnell, dafür ist ihr Anschaffungspreis aber auch noch sehr hoch.

## 1.4 Wichtige Grundbegriffe für Anfänger

Wir befassen uns jetzt mit wichtigen Grundlagen, die Sie unbedingt durchlesen sollten, bevor Sie mit CP/M arbeiten. Dabei spielt es keine Rolle, ob Sie einen CPC 464 oder 664 besitzen, denn das Arbeiten mit CP/M 2.2 ist für beide Rechner gleich.

### 1.4.1 Der CP/M-Start

Wir wissen bereits, daß die CPC-Computer nach dem Einschalten nur unter BASIC betrieben werden können. Das CP/M-Betriebssystem dagegen muß erst von Diskette nachgeladen und initialisiert werden. Falls Sie einen CPC 464 verwenden, achten Sie darauf, daß mindestens ein Diskettenlaufwerk ordnungsgemäß angeschlossen ist (s.o.). Schalten Sie zuerst das Laufwerk und dann den Computer ein, da sonst die gesamte Anlage nicht richtig funktioniert.

Beim CPC 664 haben Computer und eingebautes Floppylaufwerk nur einen gemeinsamen Schalter. Das Laufwerk ist direkt nach dem Einschalten des Computers betriebsbereit, wobei allerdings die Kabel zwischen Computer und Bildschirm-Monitor richtig eingesteckt sein müssen.

Computer bzw. Floppylaufwerk liegt beim Kauf eine Systemdiskette bei, die verschiedene Demoprogramme für BASIC, die Programmiersprache Dr. Logo und, was uns hier interessiert, das Betriebssystem CP/M 2.2 einschließlich sämtlicher Dienstprogramme enthält.

Nun legen Sie die Diskettenseite mit der Aufschrift CP/M 2.2 in das Laufwerk und geben dann

~CPM <ENTER>

ein. Durch diesen Vorgang wird CP/M geladen und gestartet.

Nun schaltet der Bildschirm auf die 80-Zeichendarstellung um und zeigt folgende Einschaltmeldung an:

```
CP/M 2.2 - Amstrad Consumer Electronics
plc A>
```

Das A mit der spitzen Klammer ist das Anforderungszeichen, oder Prompt, d.h. CP/M wartet jetzt auf Ihre Anweisungen.

### 1.4.2 Die drei Grundelemente von CP/M

Eingangs haben wir schon erfahren, das CP/M ein universelles Betriebssystem ist, das an fast jeden Computer mit einem 8080- oder Z80-Mikroprozessor angepaßt werden kann.

Wir wissen auch schon, daß ein Computer aus flüchtigem RAM-Speicher und verschiedenen ROMs sowie anderen Bausteinen besteht. Der Computer, einschließlich sämtlicher Bauteile, wird als Hardware bezeichnet. Programme, gleich welcher Art, die man von Diskette nachladen muß und die im RAM-Speicher abgelegt werden, nennt man Software.

Die Hardware des Computers enthält somit auch dessen Betriebssystem, das allerdings nicht mit dem CP/M-Betriebssystem identisch ist. Weiter oben hatten wir, um den Aufbau eines Computersystems zu erklären, der Einfachheit halber nur von einem Betriebssystem gesprochen.

Das CP/M-Betriebssystem dagegen besteht aus den drei Teilen CCP ( Command Control Processor), BDOS (Basic Disc Operation System) und BIOS (Basic Input Output System). Das Wort "Basic" hat hier jedoch nichts mit der Programmiersprache BASIC zu tun.

Beginnen wir mit dem BDOS, dem Kernstück des CP/M-Betriebssystems. Das BDOS steuert sämtliche Diskettenoperationen, es schreibt Dateien auf Diskette und liest sie wieder und verwaltet das Disketten-Inhaltsverzeichnis (Directory) . Darüber hinaus besitzt es auch Routinen zur Abfrage der Tastatur und zur Ausgabe von Zeichenketten (Strings) auf dem Bildschirm oder auf dem Drucker.

Das BDOS besteht aus geräteunabhängigen Maschinenroutinen, die für alle CP/M-Computer gleich sind. Lediglich der Speicherbereich, in dem das BDOS abgelegt ist, kann von Gerät zu Gerät unterschiedlich sein.

Das BDOS besitzt die vorteilhafte Eigenschaft, daß CP/M-Programme, die auf einem Computer entwickelt wurden, auch auf einem anderen CP/MComputer lauffähig sind. Dies wird durch genormte Funktionsaufrufe erreicht, wobei für jede Funktion eine Konstante in das C-Register des



Prozessors geladen und dann das Unterprogramm ab Adresse 5 (0005H) aufgerufen wird.

Falls Sie sich bereits in der Assemblerprogrammierung auskennen, dürfte es Ihnen nicht schwerfallen, diesen Vorgang zu verstehen. Betrachten wir ein kleines Beispiel:

Wir wollen eine Zeichenkette auf dem Bildschirm ausgeben. Dazu laden wir das C-Register mit der Konstante 9 und rufen das Unterprogramm ab der Speicherzelle 5 auf. So einfach ist das.

Für diejenigen Leser, die sich bisher nur mit BASIC befaßt haben, hier ein kleines BASIC-Programm, das die Funktionsweise der BDOS-Aufrufe simuliert.

```
 10 REM BDOS-Simulation
 20 INPUT"Funktionsnummer"; a
 30 GOSUB60
 40 GOTO 10
 50 :
60 ON a GOTO 100, 200, 300
70 :
100 REM Stringausgabe
   (
Anweisungen)
150 RETURN
190 :
200 REM Abfrage Tastatur
   (
Anweisungen)
250 RETURN
290 :
300 REM Datei auf Diskette schreiben
   (
Anweisungen)
350 RETURN
```

Nach Eingabe der Funktionsnummer, die wir hier in der Variablen a ablegen, wird das Unterprogramm in Zeile 60 aufgerufen. Dort steht die BASIC-Anweisung

```
ON a GOTO 100, 200, 300
```

die Sie vielleicht schon aus dem Handbuch oder einem BASIC-Lehrbuch kennen. Enhält die Variable a in dieser Zeile den Wert 1, wird die Routine ab Zeile 100 aufgerufen, beim Wert 2 ist es die Routine ab Zeile 200 und beim Wert 3 die entsprechende ab Zeile 300.

Die Zuordnung eines Wertes zur Variablen a entspricht dem Laden des C-Registers mit einer Funktions-Konstanten und die Abarbeitung von Zeile 60 dem jeweiligen BDOS-Aufruf in Adresse 5. BDOS verzweigt dann intern in die Routine, die mit der Funktionsnummer aufgerufen wurde.

In unserem BASIC-Programm haben wir nur drei verschiedene BDOS-Aufrufe simuliert; in Wirklichkeit kann das BDOS aber über 30 verschiedene Funktionen ausführen.

Das BDOS steht in ständigem Kontakt mit dem CCP, den man auch als Bedienungsprozessor bezeichnen kann. Ebenso wie das BDOS ist auch der CCP systemunabhängig und steuert den Dialog mit dem Benutzer. So gibt z. B. der CCP das Anforderungszeichen A> aus und wartet, daß Sie einen Befehl eingeben, den er dann analysiert und ausführt.

Eine Reihe von residenten Befehlen ist bereits im CCP enthalten. Im Gegensatz dazu gibt es auch nicht residente Befehle, zu deren Ausführung erst ein Maschinenprogramm von Diskette nachgeladen werden muß. Mehr hierüber in Kapitel 2.

Kommen wir schließlich zum BIOS, welches das Bindeglied zwischen dem BDOS und der jeweiligen Hardware darstellt. Während CCP und BDOS auf jedem CP/M-Computer identisch sind, ist dies beim BIOS nicht der Fall, denn es muß jeweils angepaßt bzw. neu geschrieben werden.

Um die BIOS-Anpassung brauchen Sie sich bei Ihrem CPC-Computer glücklicherweise nicht mehr zu kümmern, denn dies ist bereits werkseitig geschehen. Das BIOS ist hier ein fester Bestandteil des Floppy-Roms und verwendet teilweise die gleichen Routinen, die auch unter AMSDOS, dem Diskettenbetriebssystem für BASIC, Anwendung finden. Auch ist die Dateiverwaltung unter AMSDOS derjenigen unter CP/M weitestgehend angepaßt, weshalb Sie einige Dateien auch zwischen BASIC und CP/ M austauschen können. Doch hierüber später mehr.

Ähnlich wie das BDOS enthält auch das BIOS eine Tabelle mit konstanten Einsprungadressen. Die aufgerufenen BIOS-Routinen befinden sich aber auf einer weit niedrigeren Maschinenebene als die des BDOS. So ist das BDOS beispielsweise in der Lage, eine ganze Zeichenkette auf dem Bild-schirm auszugeben. Dabei ruft es wiederholt eine BIOS-Routine auf, die diese Kette jeweils zeichenweise an die Hardware weitergibt, was in diesem Fall der Bildschirm ist.

Neben den drei Hauptteilen von CP/M, CCP, BDOS und BIOS dürfen wir aber nicht vergessen, daß ein CP/M-System auch einen Arbeitsspeicher benötigt. Er wird in der Fachsprache als TPA (Transient Program Area) bezeichnet und muß einen zusammenhängenden Speicherbereich umfassen. Er beginnt bei der Adresse 100H (256 dez.) und enthält im oberen Bereich die drei Elemente CCP, BDOS und BIOS. Je größer der TPA ist, desto umfangreichere Programme können auf dem System laufen. Auf dem CPC stehen unter CP/M 2.2 ca. 41K an TPA zur Verfügung.



# 2 Allgemeine Grundlagen

Nachdem wir im ersten Kapitel wichtige Grundbegriffe über CP/M kennengelernt haben, wollen wir jetzt einen Schritt weiter gehen und die ersten praktischen Gehversuche unternehmen.

Wir wissen bereits, daß wir das CP/M-Betriebssystem zuerst laden und initialisieren müssen, bevor wir damit arbeiten zu können. Diesen Vor-gang wollen wir hier kurz rekapitulieren. Legen Sie nach dem Einschalten von Floppy und Computer Seite A der Systemdiskette in das Laufwerk A und geben Sie dann

```
CPM <ENTER>
```

ein. Falls Sie nur mit einem Laufwerk arbeiten, ist dies immer das Bezugslaufwerk bzw. Laufwerk A. Es erscheint die Einschaltmeldung und das Anforderungszeichen "A>" und Sie können mit CP/M arbeiten.

## 2.1 Einige Tippversuche

Der CCP, der den Dialog mit dem Benutzer herstellt, ist anders zu bedienen als der Editor, der unter BASIC zur Verfügung steht. In diesem Zusammenhang benötigen wir einige Steuerzeichen, die meist mit der CTRL-Taste erzeugt werden. Lassen Sie uns deshalb ein paar Übungen durchführen.

Wir beginnen mit einem ganz einfachen Befehl, der das Inhaltsverzeichnis ( Directory) der Systemdiskette auflistet. Dies funktioniert ähnlich wie unter BASIC und AMSDOS, wo wir hierzu

```
1dir
```

eingegeben haben. Unter CP/M geschieht dies mit

```
A>dir <ENTER>
```

oder

A>DIR <ENTER>

Der Vollständigkeit halber haben wir hier das Anforderungszeichen "A>" mit aufgeführt, um die gesamte Befehlszeile darzustellen. Sie dürfen es aber keinesfalls nochmals eingeben, da sonst ein Fehler auftritt. Sie müssen lediglich die drei Zeichen "D", "I" und "R" eingeben und an-schließend die ENTER-Taste drücken. In der Regel spielt es bei CP/M keine Rolle, ob Sie die Befehlszeile in Groß- oder in Kleinbuchstaben eingeben, intern wird sie ohnehin in Großschrift umgewandelt.

Wenn Sie bisher alles richtig gemacht haben, erscheint das Directory wie folgt auf dem Bildschirm:

```
A>dir
 A: MOVCPM  COM : PIP      COM  SUBMIT  COM : XSUB     COM
 A: ED      COM : ASM      COM  DDT     COM : LOAD     COM
 A: STAT    COM : DUMP     COM  DUMP    ASM : AMSDOS   COM
 A: FILECOPY COM : SYSGEN  COM  BOOTGEN COM : COPYDISC COM
 A: CHKDISC COM : DISCCOPY COM  DISCCHK COM : SETUP    COM
 A: FORMAT  COM : CSAVE    COM  CLOAD   COM : EX1      BAS
 A: EX2     BAS : ROINTIME DEM
 A>
```

Mit seinem Aufbau werden wir uns an späterer Stelle noch befassen. Im Augenblick ist es nur wichtig, daß Sie die Befehlsübergabe richtig beherrschen.

Beachten Sie, daß nach Ausgabe des Directories wieder das Anforderungszeichen "A>" und der Cursor erscheint. CP/M ist jetzt zur Übernahme weiterer Befehle bereit.

Bisher sind wir davon ausgegangen, daß Sie alles richtig eingegeben haben. Was geschieht aber, wenn Sie sich vertippen? Lassen Sie uns dies anhand dieses einfachen Befehls einmal ausprobieren, und geben Sie dazu als nächstes folgende fehlerhafte Anweisung ein:

A>dirr<ENTER>

Diesen Befehl versteht CP/M nicht und gibt deshalb folgende Meldung aus:

A>dirr
DIRR? **A>**

Die fehlerhafte Anweisung erscheint in Großschrift mit angefügtem Fragezeichen und das Anforderungszeichen ist wieder sichtbar, d.h. CP/M kann einen neuen Befehl entgegennehmen.



Dies heißt aber noch lange nicht, daß es überhaupt soweit kommen muß, insbesondere dann nicht, wenn Sie den Fehler noch vor dem Drücken der ENTER-Taste bemerken. In diesem Fall können Sie die DEL-Taste drücken, wodurch das letzte Zeichen gelöscht wird und der Cursor um eine Stelle nach links rückt. Da der Befehl jetzt richtig geschrieben ist, kann er nun auch durch Drücken der ENTER-Taste ausgeführt werden.
Betrachten wir ein weiteres Beispiel:

A>fir

Hier wurde nicht das letzte, sondern bereits das erste Zeichen falsch eingegeben. Um diesen Fehler zu korrigieren, haben wir zwei Möglichkeiten:

Zunächst einmal können wir dreimal hintereinander die DEL-Taste drücken, wodurch die gesamte Zeile gelöscht wird. Da beim CPC sämtliche Tasten mit einer Autorepeat-Funktion ausgestattet sind, brauchen wir die DEL-Taste nur lange genug zu drücken, bis alle drei Zeichen verschwunden sind. Dabei werden Sie jedoch feststellen, daß das Anforderungszeichen nicht gelöscht werden kann.

Es gibt aber noch eine andere Funktion, die die gesamte Befehlszeile auf einmal löscht. Halten Sie dazu die CTRL-Taste gedrückt und drücken Sie anschließend die X-Taste. Einen solchen kombinierten Tastendruck beschreibt man mit

CTRL—X

oder kürzer

^X

In manchen Fällen erscheint die letzte Schreibweise auch auf dem Bildschirm, da die mit der CTRL-Taste erzeugten Steuerzeichen sonst nicht dargestellt werden können.

Nachdem Sie nun die Zeile gelöscht haben, geben Sie sie nochmals richtig ein und drücken die ENTER-Taste. Beachten Sie, daß Sie eine fehlerhafte Befehlszeile immer soweit löschen müssen, bis alle falschen Zeichen entfernt sind. Von dieser Stelle an geben Sie dann die richtigen Zeichen ein. Es ist also nicht in jedem Fall erforderlich, die gesamte Zeile zu löschen.

Außer <CTRL—X> gibt es aber noch eine Reihe anderer <CTRL>-Steuerzeichen, von denen wir die wichtigsten hier behandeln werden. Außerdem befindet sich im Anhang eine Zusammenfassung sämtlicher Steuerzeichen.

Doch nun noch einmal zurück zur DEL-Taste. Statt <DEL> könnten wir nämlich auch <CTRL—H> eingeben, was die gleiche Wirkung hat. hnliches gilt auch für die ENTER- (RETURN-) Taste, die wir durch <CTRL¯M> ersetzen können.

Falls Sie einmal eine sehr lange Befehlszeile eingeben müssen, die nicht in eine Bildschirmzeile paßt, können Sie sich folgendermaßen behelfen: Nach Abschluß einer jeden Bildschirmzeile drücken Sie <CTRL—E> (nicht <ENTER! >), worauf Sie in der nächsten Zeile weiterschreiben können. Erst wenn Sie Ihren Befehl komplett eingegeben haben, drücken Sie die ENTER-Taste.

Als kleines Demonstrationsbeispiel wollen wir einmal den DIR-Befehl auf drei Bildschirmzeilen verteilen, wobei jede Zeile nur ein Zeichen enthält:

A>D<CTRL–E>
I<CTRL–E>
R<ENTER>

Auch in diesem Fall wird das Directory genauso wie vorher aufgelistet. Übrigens wollen wir von nun an sämtliche CP/M-Befehle groß schreiben, um sie besser hervorzuheben. Dies ist jedoch, wie wir oben bereits gesehen haben, keinesfalls notwendig.

Wir werden in diesem Buch noch viele Befehle kennenlernen, die etwas auf dem Bildschirm auflisten, wie es auch beim DIR-Befehl der Fall ist. Häufig ist das Listing so umfangreich, daß es nicht auf den Bildschirm paßt und sich infolgedessen nach oben schiebt. Diesen Vorgang nennt man auch Scrollen. Hier können wir mit <CTRL–S> den Listvorgang unterbrechen und durch Drücken einer beliebigen anderen Taste fort-setzen. Versuchen Sie dies einmal, während Sie das Directory listen.

Unter CP/M können Sie von der hervorragenden Eigenschaft Gebrauch machen, daß alles, was auf dem Bildschirm gelistet wird, gleichzeitig über den Drucker ausgegeben wird. Achten Sie aber unbedingt darauf, daß Ihr Drucker auch angeschlossen und eingeschaltet ist. CPC-Computer haben nämlich die unangenehme Eigenschaft, in einer Endlosschleife zu hängen, wenn der Drucker angesprochen, aber nicht angeschlossen ist.

Ist der Drucker betriebsbereit, können Sie ihn mit <CTRL–P> der Bildschirmausgabe parallel schalten. Probieren Sie das einmal aus und listen Sie dann ganz normal das Directory. Sie erhalten den Inhalt Ihrer Diskette damit schwarz auf weiß ausgedruckt. Durch eine erneute Eingabe von <CTRL¯P> schalten Sie anschließend den Drucker wieder ab.



Abschließend wollen wir auch in diesem Zusammenhang <CTRL¯C> nicht vergessen, das ein gerade ablaufendes Programm unterbricht und einen Warmstart auslöst, wobei CCP und BDOS neu geladen werden. Dieser Befehl sollte auch nach jedem Diskettenwechsel ausgeführt wer-den, um Fehler zu vermeiden, die unter Umständen zu einer Zerstörung der Dateien auf der Diskette führen können.

## 2.2 Etwas über Dateien

Ebenso wie bei BASIC gibt es auch bei CP/M Dateien. Allgemein ausgedrückt ist eine Datei eine Ansammlung von Daten, die auf Diskette oder einem anderen Datenträger abgelegt sind. Jede Datei erhält einen Namen, der in das Inhaltsverzeichnis (Directory) der Diskette eingetragen wird.

Nun gibt es allerdings eine Vielzahl von verschiedenen Dateien, welche die unterschiedlichsten Arten von Informationen enthalten können. So gibt es z.B. Programmdateien, die ein ausführbares Programm enthalten, Textdateien, die von einem Textverarbeitungssystem (z.B. WORDSTAR) erzeugt werden oder reine Datendateien, in denen beispielsweise ein Adressenverzeichnis abgelegt ist. Selbst wenn Sie Ihre Farbgraphiken auf Diskette schreiben, legen Sie eine Datei an, die in diesem Fall den Inhalt des Bildschirm-RAMs speichert.

In diesem Abschnitt erhalten Sie einen ersten Einblick in die verschiedensten Dateiarten, die unter CP/M eine Rolle spielen. Nachdem wir im letzen Abschnitt bereits das Directory der Systemdiskette aufgelistet haben, wollen wir dies jetzt nochmals wiederholen und uns die Einträge darin einmal genauer ansehen.

Von nun an verzichten wir darauf, das Anforderungszeichen "A>" mit anzugeben und beschränken uns lediglich auf die einzugebende Befehlszeile.

Hier noch einmal das Directory:

```
DIR
A: MOVCPM   COM : PIP      COM : SUBMIT   COM : XSUB     COM
A: ED       COM : ASM      COM : DDT      COM : LOAD     COM
A: STAT     COM   DUMP     COM : DUMP     ASM : AMSDOS   COM
A: FILECOPY COM   SYSGEN   COM : BOOTGEN  COM : COPYDISC COM
A: CHKDISC  COM   DISCCOPY COM : DISCCHK  COM : SETUP    COM
A: FORMAT   COM : CSAVE    COM : CLOAD    COM : EX1      BAS
A: EX2      BAS : ROINTIME DEM
```

Mit dem DIR-Befehl werden sämtliche Dateinamen in der Reihenfolge ausgegeben, in der sie im Directory eingetragen sind. Dabei erscheinen jeweils vier Namen nebeneinander in einer Zeile. Am Zeilenanfang ist das Laufwerk aufgeführt, von dem das Directory gelistet wurde. In unserem Fall ist es Laufwerk A.

Falls Sie ein zweites Laufwerk B angeschlossen haben, können Sie auch das Directory von dessen Diskette listen, was mit dem Befehl

```
DIR B:
```

geschieht. Achten Sie darauf, daß zwischen den Bezeichnungen DIR und B: ein Leerzeichen frei bleibt, da sonst CP/M den Befehl falsch interpretiert. Diese Schreibweise gilt übrigens für die meisten CP/M-Befehle, falls sie sich auf ein anderes als das Bezugslaufwerk beziehen.

Normalerweise ist immer Laufwerk A das Bezugslaufwerk. Sie können allerdings auch Laufwerk B dazu bestimmen, indem Sie

```
B:
```

eingeben. Wenn Sie jetzt den DIR-Befehl erteilen, erhalten Sie das Directory von Laufwerk B. Das Inhaltsverzeichnis von A erscheint in diesem Fall entsprechend mit

```
DIR A:
```

Wollen Sie später wieder A zum Bezugslaufwerk bestimmen, geben Sie lediglich

```
A:
```

ein. Für den Normalfall wollen wir aber Laufwerk A als Bezugslaufwerk beibehalten und nur in Ausnahmefällen auf B zurückgreifen. Außerdem sollte immer Laufwerk A die Diskette mit den Systemspuren enthalten, in denen u.a. auch der CCP und das BDOS abgelegt ist. Deshalb wird auch bei jedem Warmstart infolge von <CTRL—C> auf Laufwerk A zugegriffen, selbst wenn B zum Bezugslaufwerk erklärt wurde. Mit den System-spuren werden wir uns an anderer Stelle noch ausführlich befassen.

Betrachten wir das aufgelistete Directory, stellen wir fest, daß jeder Dateiname einen Zusatz enthält, der als Dateityp oder Extension bezeichnet wird. Wie Sie sehen, ist der überwiegende Teil der Dateinamen mit dem Zusatz COM versehen.

COM steht für englisch "Conimand" und kennzeichnet unter CP/ M eine Befehlsdatei. Solche Dateien haben die Eigenschaft, daß sie nach dem



Aufruf sofort ausgeführt werden. Dazu müssen Sie nur ihren Namen (ohne COM) als Befehlszeile eingeben und anschließend die ENTER-Taste drücken.

Die Dateinamen EX1 und EX2 enthalten den Zusatz BAS und sind reine BASIC-Dateien, die Sie nicht unter CP/M ausführen können. Ähnliches gilt für ROINTIME mit der Extension DEM. Hierbei handelt es sich um ein Spielprogramm, das nur von BASIC aus mit

```
RUN "ROINTIME.DEM"
```

gestartet werden kann.

Sie sehen also, daß CP/M- und BASIC-Programme auf ein und derselben Diskette untergebracht sein können und intern vom Disketten-Betriebssystem (DOS) ähnlich verwaltet werden. Unter gewissen Umständen können und müssen BASIC-Programme sogar mit CP/M-Dienstprogrammen bearbeitet werden. Wir werden uns noch näher mit den CP/M-Programmen befassen, die einzelne Dateien oder ganze Disketten kopieren. Dabei spielt es keine Rolle, ob es sich um BASIC-Dateien, die unter AMSDOS arbeiten, oder um reine CP/M- oder sonstige Dateien handelt. Selbst wenn Sie nur mit BASIC arbeiten, kommen Sie nicht umhin, einige CP/M-Dienstprogramme mit zu verwenden. Neben den erwähnten Kopierprogrammen gilt dies ganz besonders auch für das Programm FORMAT.COM, das Sie zum Formatieren von Disketten benötigen.

Dateinamen dürfen maximal acht Zeichen umfassen, an die sich die Extension von drei Zeichen anschließt. Im Gegensatz zur Auflistung des Directories mit dem DIR-Befehl muß meistens zwischen Dateinamen und Extension ein Punkt gesetzt werden, besonders wenn die Extension mit in der Befehlszeile auftaucht. Abgesehen von der Ausführung der COM-Dateien ist diese Schreibweise sogar meist zwingend. Betrachten wir ein Beispiel:

Auf der Systemdiskette befindet sich das Programm FILECOPY mit der Extension COM, also eine reine CP/M-Befehlsdatei zum Kopieren einzelner Dateien. **Im** Directory-Listing **ist** sie **mit**

```
FILECOPY COM
```

aufgeführt, wobei FILECOPY die maximale Länge von acht Zeichen umfaßt. Zwischen FILECOPY und der Extension steht ein Leerzeichen. Wäre der Dateiname weniger als acht Zeichen lang, würde sich der Abstand zur Extension entsprechend vergrößern. So umfaßt der Name PIP beispielsweise nur drei Zeichen, weshalb er folgendermaßen mit dem DIR-Befehl ausgegeben wird:

```
PIP      COM
FILECOPY COM
```

Zum Vergleich haben wir nochmals den Namen FILECOPY darunter gesetzt. Da es sich in beiden Fällen um COM-Dateien handelt, werden sie mit

```
PIP <ENTER>
```

bzw.

```
FILECOPY <ENTER>
```

aufgerufen und ausgeführt. Von diesem Sonderfall einmal abgesehen, werden beide Namen zusammen mit ihrer Extension als

```
PIP.COM
```

bzw.

```
FILECOPY.COM
```

geschrieben, wobei zwischen Dateinamen und Extension lediglich ein Punkt steht. Diese Schreibweise wollen wir auch im vorliegenden Buch beibehalten. Nachfolgend einige Beispiele für generell zulässige Dateinamen:

```
TEST.COM
BUCH.TXT
WS.COM
BIBLTHEK.DAT
LAGER.  BAK
PROBE.  BAS
ERW.BIN
TITEL.
```

Nicht alle hier verwendeten Extensionen haben eine bestimmte Bedeutung und können teilweise auch frei gewählt bzw. weggelassen werden. Dies ist besonders bei reinen Daten- oder Textdateien häufig der Fall.

In der Praxis sind jedoch einige Extensionen für bestimmte Zwecke reserviert, die Sie bei der Wahl der Dateinamen berücksichtigen sollten:



| | |
|---|---|
| COM | Befehlsdatei |
| BAK | Backup- (Sicherungs-) Datei |
| SUB | Stapelverarbeitungsdatei |
| ASM | Datei mit Assembler-Quelltext |
| BAS | BASIC-Programm |
| BIN | Binärdatei unter BASIC bzw. AMSDOS |
| PRN | Datei für Listenausdruck assemblierter Maschinenprogramme |
| HEX | Datei im INTEL-HEX-Format |
| $$$ | Zwischendatei (nur für bestimmte Anwendungen) |

Käufliche BASIC- und CP/M-Programme verwenden auch noch weitere Extensionen, die gegebenenfalls zu beachten sind.

Der DIR-Befehl eignet sich nicht nur zum Auflisten des gesamten Directories, sondern kann auch zur Abfrage einzelner oder mehrdeutiger Dateinamen verwendet werden.

Angenommen, Sie möchten feststellen, ob auf Ihrer Diskette die Datei FILECOPY.COM vorhanden ist, ohne erst das gesamte Directory durchsuchen zu müssen. Dazu geben Sie lediglich

```
DIR FILECOPY.COM
```

ein. Dies ist ein Anwendungsfall, in dem der Dateiname mit der durch einen Punkt abgetrennten Extension anzugeben ist. Ist nun FILECOPY.COM auf der Diskette enthalten, erscheint die Ausgabe

```
A: FILECOPY COM
```

in der gleichen Schreibweise wie beim einfachen DIR-Befehl. Wurde die gewünschte Datei dagegen nicht gefunden, wird die Meldung

```
NO FILE
```

ausgegeben.

In diesem Beispiel handelt es sich um einen eindeutigen Dateinamen, der vorgegeben wurde. Es besteht aber auch die Möglichkeit, mehrdeutige Dateinamen anzugeben, wozu die Zeichen "*" und "?" dienen. Hier ein paar Beispiele, die sich wieder auf Dateien der Systemdiskette beziehen.

Wir haben bereits festgestellt, daß sich auf der Systemdiskette BASIC-Programme mit der Extension BAS befinden. Indem wir

```
DIR *.BAS
```

eingeben, können wir diese einzeln auflisten. Es erscheint

```
A: EX1     BAS : EX2    BAS
```

Selbstverständlich funktioniert dies genauso mit allen anderen Extensionen, so daß wir beispielsweise auch sämtliche COM-Dateien auflisten könnten.

Nun möchten wir wissen, welche COM-Dateien sich auf der Diskette befinden, die mit dem Buchstaben D beginnen. Dazu geben wir DIR D????? ??.COM
oder

DIR D*.COM

ein und erhalten die Auflistung

```
A: DDT     COM : DUMP   COM : DISCCHK COM
```

Wir sehen also, daß ein Fragezeichen einen einzelnen Buchstaben ersetzt. Ein Sternchen dagegen steht entweder für den gesamten Dateinamen bzw. die gesamte Extension oder ersetzt die restlichen Zeichen des Dateinamens, wenn die ersten fest vorgegeben sind. Um dies zu verdeutlichen, wollen wir nochmals das gesamte Directory ausgeben, diesmal allerdings mit Hilfe mehrdeutiger Dateinamen. Sämtliche nachfolgend aufgeführten Befehle haben die gleiche Wirkung:

DIR
DIR *.*
DIR ????????,*
DIR *.???
DIR ????????.???

Natürlich wird man in der Praxis hierfür nur DIR verwenden. Es gibt aber viele Befehle, die auf jeden Fall einen ein- oder mehrdeutigen Dateinamen erfordern.

Auf Ihrer Systemdiskette befindet sich auch STAT.COM, ein Programm, mit dem man vielerlei Angaben zum verwendeten CP/M-System abfragen kann. Obwohl wir STAT an späterer Stelle noch ausführlich behandeln werden, wollen wir hier etwas vorgreifen und mit Hilfe dieses Programms das Directory nochmals auflisten, diesmal allerdings in einer ausführlicheren Form als mit dem Befehl DIR.

Da STAT eine COM-Datei ist, wird zum Laden und Starten keine Extension benötigt. Wenn Sie demgemäß



STAT

eingeben, erscheint, auf die Systemdiskette bezogen,

```
A: R/W, Space: 50k
```

Diese Meldung besagt, daß auf der Diskette in Laufwerk A noch 50k zum Lesen und Schreiben (R/W) frei sind.

STAT kann aber, wie viele andere CP/M-Dateien auch, weitere Parameter übernehmen. Ein solcher Parameter kann aus einem ein- oder mehrdeutigen Dateinamen bestehen. Indem wir nun *.* für den Dateinamen vorgeben, erhalten wir sämtliche Dateinamen mit einigen Zusatzangaben aufgelistet. Nach Eingabe von

STAT *.*

erscheint nun für die Systemdiskette:

```
Recs  Bytes  Ext Acc
2      1k    1 R/W A:AMSDOS.COM
64     8k    1 R/W A:ASM.COM
10     2k    1 R/W A:BOOTGEN.COM
19     3k    1 R/W A:CHKDISC.COM
15     2k    1 R/W A:CLOAD.COM
21     3k    1 R/W A:COPYDISC.
14     2k    1 R/W A:CSAVE.COM
38     5k    1 R/W A:DDT.COM
19     3k    1 R/W A:DISCCHK.COM
21     3k    1 R/W A:DISCCOPY.
33     5k    1 R/W A:DUMP.ASM
4      1k    1 R/W A:DUMP.COM
52     7k    1 R/W A:ED.COM
9      2k    1 R/W A:EXI.BAS
3      1k    1 R/W A:EX2.BAS
22     3k    1 R/W A:FILECOPY.
21     3k    1 R/W A:FORMAT.COM
14     2k    1 R/W A:LOAD.COM
76    10k    1 R/W A:MOVCPM.COM
58     8k    1 R/W A:PIP.COM
208   26k    2 R/W A:ROINTIME.
61     8k    1 R/W A:SETUP.COM
41     6k    1 R/W A:STAT.COM
10     2k    1 R/W A:SUBMIT.COM
12     2k    1 R/W A:SYSGEN.COM
6      1k    1 R/W A:XSUB.COM
Bytes Remaining On A: 50K
```

Sämtliche Dateinamen werden - in alphabetischer Reihenfolge sortiert - aufgelistet, ähnlich wie es auch mit dem CAT-Befehl unter BASIC geschieht. Betrachten wir einmal die einzelnen Angaben von links nach rechts:

Die Spalten *Recs, Bytes* und *Ext* enthalten Angaben über den Speicher-platz, den die jeweilige Datei auf der Diskette belegt. Um sie zu verstehen, werfen wir einen kurzen Blick in die interne Diskettenorganisation, wie sie unter CP/M, aber auch unter AMSDOS Verwendung findet.

Das Betriebssystem CP/M ist so ausgelegt, daß es an die verschiedensten Diskettenformate angepaßt werden kann. Aus dem ersten Kapitel wissen wir bereits, daß das BIOS die Schnittstelle zur Hardware darstellt und für jeden CP/M Computer verschieden ist.

Nun muß jede Diskette formatiert werden, bevor darauf geschrieben wer-den kann. Dabei wird sie in eine bestimmte Anzahl von Spuren eingeteilt, die wieder in Sektoren unterteilt sind.

Stellen Sie sich einmal eine runde Torte vor, auf die eine Reihe von Rin-gen aus Sahne oder Zuckerguß aufgespritzt wurde. Schneiden wir nun die Torte in einzelne, gleichmäßige Stücke, so enthält jedes Stück immer noch einen Teil von jedem Ring.

Diesen bildlichen Vergleich können wir auch auf die Diskette übertragen. Bei der Formatierung erhält sie allerdings statt der Zuckerringe Spuren und statt der Teilringe auf den Tortenstücken Sektoren.

Auf eine formatierte CPC-Diskette sind 40 Spuren mit je 9 Sektoren aufgetragen, von denen jeder 512 Bytes an Informationen aufnehmen kann. Insgesamt ergibt dies eine Speicherkapazität von 180K, wovon noch ein kleiner Teil für die Systemspuren und das Directory abgeht, der für den Anwender nicht nutzbar ist.

Nun verwenden allerdings die meisten Computerhersteller ihr eigenes Diskettenformat mit einer unterschiedlichen Anzahl von Spuren, Sektoren und Bytes pro Sektor. Da CP/M aber auf sämtlichen Systemen lauffähig sein soll, ist es die Aufgabe des BIOS, die jeweilige Anpassung vorzunehmen.

Das CP/M-Betriebssystem arbeitet allerdings nicht mit den physikalischen Sektoren, sondern überläßt dies dem jeweiligen BIOS. Es kann nur in so-genannten logischen Records und Blöcken "denken". Ein Record ist eine Aufzeichnungseinheit von 128 Bytes, wogegen ein Block meist 1024 Bytes ( 1K) umfaßt, wie es auch beim CPC der Fall ist.

Wenn Sie eine Datei auf Diskette schreiben, belegt sie je nach Umfang eine bestimmte Anzahl von Records, die in der ersten Spalte der Auflistung angegeben ist. Das Directory kann Dateien jedoch nur blockweise verwalten, weshalb die zweite Spalte die Anzahl der Blöcke bzw. Kbytes enthält.



Zwischen Records und Blöcken besteht folglich ein logischer Zusammen-hang, denn ein Block enthält exakt acht Records. Für jede angefangenen acht Records muß also ein weiterer Block auf der Diskette reserviert wer-den, der im Directory eingetragen wird. Wenn Sie in der Auflistung die jeweilige Recordanzahl durch acht dividieren, können Sie dies leicht nachprüfen.

Selbst wenn eine Datei nur aus drei Bytes besteht, belegt sie einen Record, für den ein ganzer Block reserviert werden muß.

Die dritte Spalte gibt die Anzahl der Extents oder Erweiterungseinträge im Directory an. Extent darf aber keinesfalls mit Extension (Dateityp) verwechselt werden! Wird nun eine Datei angelegt, die mehr als 16K umfaßt, muß für jede angefangenen 16K ein weiterer Eintrag (Extent) reserviert werden, der allerdings beim Auflisten des Directories nicht gesondert erscheint. Auf der Systemdiskette benötigt ROINTIME.DEM mit 26K als einzige Datei mehr als 16K und belegt somit zwei Extents.

Beim CPC kann das Directory einer Diskette maximal 64 Einträge aufnehmen, wobei allerdings sämtliche Extents mit eingeschlossen sind.

Hinter dem Extent gibt R/W (Read/Write) an, daß auf die betreffende Datei sowohl Schreib- als auch Lesezugriffe zulässig sind. Ist eine Datei dagegen schreibgeschützt, erscheint statt dessen der Hinweis R/O (Read Only). Vor dem eigentlichen Dateinamen, der hier im Gegensatz zum DIR-Befehl in Punktschreibweise angegeben ist, ist noch das Laufwerk mit der Diskette aufgeführt, die die betreffende Datei enthält.

Ähnlich wie beim DIR-Befehl ist es auch bei STAT keinesfalls notwendig, das gesamte Directory auszulisten, denn hier gelten die gleichen Regeln für ein- und mehrdeutige Dateinamen wie oben beschrieben. So erhalten Sie beispielsweise mit

```
STAT *.COM
```

die gleichen Angaben für sämtliche COM-Dateien oder mit

```
STAT DDT.COM
```

nur für die Datei DDT.COM. Ist die gesuchte Datei nicht vorhanden, er-scheint die Meldung

```
Fite Not Found
```

Der Befehl

```
STAT B:*.*
```

listet sämtliche Dateien von Laufwerk B, (falls vorhanden), wobei die Diskette mit der STAT.COM-Datei in A verbleibt. Dieser Befehl ist immer dann nützlich, wenn Sie mit zwei Laufwerken arbeiten, wobei Sie in Laufwerk A die Systemdiskette und in Laufwerk B die zu untersuchende Diskette legen. Haben Sie dagegen kein zweites Laufwerk zur Verfügung, sollten Sie das STAT.COM-Programm zunächst auf die zu untersuchende Diskette kopieren, was z.B. mit FILECOPY (s.u.) geschehen kann.

## 2.3 Disketten formatieren

Was bei der Formatierung geschieht, haben wir bereits im letzten Ab-schnitt kennengelernt. Hier wollen wir nun die Formatierung in der Praxis durchführen.
Legen Sie zunächst die Systemdiskette in Laufwerk A und geben Sie

```
FORMAT
```

ein, womit Sie das Programm FORMAT.COM laden und starten. Befolgen Sie nun die Anweisung

```
Please insert disc to be formatted into
drive A then press any key:

(Bitte zu formatierende Diskette in A
einlegen, dann beliebige Taste drücken:)
```

Nehmen Sie die Systemdiskette wieder heraus und legen Sie jetzt die Diskette ein, die formatiert werden soll. Dann drücken Sie eine beliebige Taste, wodurch die Formatierung eingeleitet wird.

Achtung! Formatieren Sie nur fabrikneue Disketten oder solche, auf deren Inhalt Sie unbedingt verzichten können. Durch die Formatierung werden nämlich alle eventuell gespeicherten Daten unwiderruflich zerstört!

Die 3-Zoll-CPC-Disketten können Sie beidseitig benutzen und somit auch beidseitig formatieren (Seite A und B). Im Gegensatz zu einigen anderen Diskettenlaufwerken können beide Seiten aber nur unabhängig voneinander eingesetzt werden. Sie erhalten somit lediglich eine insgesamt doppelt so große Speicherkapazität.

Die eben beschriebene Formatierungsanweisung ist für das Arbeiten mit CP/M in der Regel völlig ausreichend. Strenggenommen wird die Diskette hier nämlich nicht nur formatiert, sondern gleichzeitig mit dem CP/M-



Betriebssystem versehen, das in den äußersten beiden Spuren 0 und 1 abgelegt wird. Sie kann also ohne weitere Präparierung zum Arbeiten mit CP/M eingesetzt werden, da sie bereits das Systemformat enthält.

Außer dem Systemformat gibt es aber noch drei andere Formate, in denen Sie Disketten formatieren können. Dazu ist hinter FORMAT noch ein Parameter anzugeben. So formatiert

`FORMAT D` im Datenformat,
`FORMAT V` im Vendor-Format und
`FORMAT 1` im IBM-Format

Zwischen dem Befehl FORMAT und dem Parameter muß jeweils ein Leerzeichen stehen.

Eine im Datenformat formatierte Diskette enthält keine Systemspuren und umfaßt deshalb eine etwas größere Speicherkapazität. Unter CP/M können Sie sie nur in Laufwerk B einsetzen, da auf die Systemspuren immer nur in Laufwerk A zugegriffen wird. Auch wenn Sie in BASIC arbeiten, benötigen Sie die CP/M-Systemspuren nicht und können das Datenformat verwenden.

Das Vendor-Format dagegen verwendet man für solche Disketten, die professionell angebotene Software enthalten. Die Formatierung findet hier genauso wie beim Systemformat statt, jedoch werden die beiden äußeren Systemspuren nur reserviert, aber nicht mit dem CP/M-Betriebssystem versehen. Der Anwender muß es dann selbst in die Systemspuren schreiben, da es neben der eigentlichen Software nicht mitverkauft wer-den darf. Zum Übertrag des CP/M-Betriebssystems dienen die Programme BOOTGEN. COM und SYSGEN.COM, die an späterer Stelle noch erläutert werden.

Im IBM-Format müssen eigentlich nur solche Disketten formatiert wer-den, die zum Datenaustausch mit anderen Rechnern dienen, die dieses Format ebenfalls lesen können. Allerdings geschieht dies meist mit 5.25-Zoll- und nicht mit den 3-Zoll-Minidisketten des CPC. Die einzige realistische Einsatzmöglichkeit besteht nur dann, wenn man ein 5.25-Zoll-Laufwerk als Laufwerk B anschließt, das von verschiedenen Herstellern für den CPC angeboten wird. Dieses Laufwerk sollte jedoch den gleichen Controller verwenden wie das 3-Zoll-Laufwerk A.

Da FORMAT.COM nur Disketten in Laufwerk A formatieren kann, muß man sich eines kleinen Tricks bedienen. Zunächst formatiert man eine 3-Zoll-Diskette in Laufwerk A im IBM-Format und kopiert diese mit COPYDISC.COM (s.u.) auf die 5.25-Zoll-Diskette in Laufwerk B. Selbst

wenn diese Diskette keine Daten enthält, so werden doch die IBM-Spuren und -Sektoren übertragen. Nun kann man beliebige Dateien ganz normal auf Laufwerk B schreiben oder dorthin kopieren, beispielsweise mit dem Programm PIP.COM, das wir noch ausführlich behandeln werden.

Das Lesen von IBM-formatierten 5.25-Zoll-Disketten kann dagegen wie gewöhnlich von Laufwerk B aus stattfinden, denn der Controller erkennt automatisch das verwendete Format.

## 2.4 Disketten kopieren

Nachdem wir uns mit der Formatierung befaßt haben, wollen wir jetzt daran gehen, eine Sicherheitskopie von der Systemdiskette herzustellen. Dies sollten Sie übrigens auch mit allen anderen Disketten tun, die wichtige Daten enthalten. Hin und wieder kommt es nämlich vor, daß die Diskette, aus welchem Grund auch immer, beschädigt oder sogar zerstört wird. In solchen Fällen ist man dann froh, wenn man noch eine zweite Kopie besitzt.

Derartig kopierte Disketten nennt man Arbeitsdisketten, und zwar des-halb, weil man sie zum Arbeiten benutzt, während die Orginaldiskette an einem sicheren Ort aufbewahrt werden sollte. Dies wollen wir auch mit unserer Systemdiskette so handhaben und fertigen deshalb eine Kopie an.

Auf der Systemdiskette selbst befinden sich zwei Programme, die zum Kopieren von ganzen Disketten dienen. Wenn Sie sich das Directory ansehen, finden Sie dort die beiden Dateien

DISCCOPY.COM (für ein Laufwerk) und
COPYDISC.COM (für 2 Laufwerke)

Auch hierbei handelt es sich um COM-Dateien, die Sie folgendermaßen laden und starten müssen:

DISCCOPY <ENTER> bzw.

COPYDISC <ENTER>

Bei der Verwendung von zwei Laufwerken (COPYDISC) ist das Kopieren relativ einfach. Sie legen die Quelldiskette (engl. source disc: Diskette,die kopiert werden soll) in Laufwerk A und die Zieldiskette (engl. destination



disc: Diskette, auf die kopiert werden soll) in Laufwerk B und drücken dann eine beliebige Taste, wodurch der Kopiervorgang gestartet wird. Im Anschluß daran werden Sie gefragt, ob Sie eine weitere Diskette kopieren möchten, wobei Sie mit Y (ja) bzw. N (nein) antworten müssen. Bei der Zieldiskette spielt es keine Rolle, ob sie bereits formatiert ist oder nicht, da dies automatisch während des Kopiervorganges geschieht.

Haben Sie allerdings nur ein Laufwerk zur Verfügung, müssen Sie das Programm DISCCOPY verwenden. Es funktioniert ähnlich wie COPYDISC, jedoch müssen Sie mehrfach die Quell- und Zieldiskette austauschen. Hierbei wird jeweils ein Teil der Quelldiskette gelesen, im Arbeitsspeicher abgelegt und nach dem Diskettenwechsel auf die Zieldiskette geschrieben. Die Diskette muß immer dann gewechselt werden, wenn die Meldung

```
Please insert source disc into drive A and press any
key: (Bitte Duelldiskette in A legen und beliebige
Taste drücken)
```

bzw.

```
Please insert destination disc into drive A and press any
key: (Bitte Zieldiskette in A legen und beliebige Taste
drücken)
```

erscheint.

Neben diesen beiden Kopierprogrammen gibt es noch die beiden Dateien

DISCCHK.COM (für ein Laufwerk) und
CHKDISC.COM (für zwei Laufwerke)

mit deren Hilfe überprüft werden kann, ob eine Diskette fehlerfrei kopiert wurde. Die Bedienung dieser beiden Programme ist den Kopierprogrammen sehr ähnlich, wobei Sie bei der Verwendung von einem Laufwerk mehrmals die Disketten wechseln müssen.

## 2.5 Einzelne Dateien kopieren

Auf der Systemdiskette befindet sich das Programm FILECOPY.COM, mit dem Sie einzelne Dateien von einer Diskette auf eine andere kopieren können, wo sie unter ihrem ursprünglichen Namen abgelegt wird. Das Programm ist so ausgelegt, daß es mit einem Laufwerk arbeitet. Je nach Größe der zu kopierenden Datei müssen dabei Quell- und Zieldiskette

ein- oder mehrmals gewechselt werden, ähnlich wie es bei DISCCOPY der Fall ist.

Angenommen, wir wollen eine Datei kopieren, die unter dem Namen LAGER.DAT auf Diskette abgelegt ist. Dazu geben wir folgendes ein:

```
FILECOPY LAGER.DAT
```

Achten Sie darauf, daß hinter FILECOPY immer der Name der zu kopierenden Datei mit Extension angegeben ist, selbst wenn sie sich nicht auf der gleichen Diskette wie FILECOPY befindet. Dann werden Sie, ähnlich wie bei DISCCOPY, aufgefordert, die Quelldiskette, auf der sich die betreffende Datei befindet, einzulegen und eine beliebige Taste zu drücken. Dasselbe wiederholt sich mit der Zieldiskette, auf die die Datei unter gleichem Namen geschrieben werden soll.

Die Zieldiskette muß allerdings formatiert sein, wobei es aber keine Rolle spielt, in welchem Format dies geschehen ist. Auf diese Weise kann man leicht Dateien von einem Diskettenformat auf ein anderes übertragen.

Hinter FILECOPY kann man auch mehrdeutige Dateinamen angeben. So kopiert z.B.

```
FILECOPY *.COM
```

sämtliche COM-Dateien und

```
FILECOPY *.*
```

sämtliche Dateien überhaupt, die sich auf der Quelldiskette befinden.

Denjenigen Lesern, die sich bereits mit CP/M auskennen, sei gesagt, daß FILECOPY teilweise ähnliche Eigenschaften besitzt, wie das CP/M Standard-Dienstprogramm PIP, das wir an späterer Stelle noch ausführlich behandeln werden. FILECOPY ist, wie auch DISCCOPY, COPYDISC, DISCCHK, CHKDISC, FORMAT und BOOTGEN ein Programm, das speziell auf die Hardware des CPC abgestimmt ist und keinesfalls zur Standardausrüstung anderer CP/M-Systeme gehört, jedenfalls nicht in dieser Form.

Selbstverständlich kann man auch mit PIP Dateien wie mit FILECOPY kopieren, jedoch braucht man hier in jedem Fall zwei Laufwerke, während FILECOPY nur eines benötigt. FILECOPY kommt vor allem den Anwendern Anwendern zugute, die nur ein Laufwerk zur Verfügung haben.



## 2.6 DISCKIT2

Das Programm DISCKIT2 steht normalerweise nur dem CPC6128-Anwender zur Verfügung und wird auch nur mit diesem Computer zusammen geliefert. Wir wollen es hier aber trotzdem behandeln, da es einerseits auch auf dem CPC 464 und 664 läuft und andererseits sehr komfortabel in der Bedienung ist. Es ist die CP/M 2.2-Version von DISCKIT3, das für CP/M Plus auf dem CPC 6128 entwickelt wurde. DISCKIT2 ist menügesteuert und umfaßt sämtliche Routinen, die zum Formatieren, Kopieren und Verifizieren von Disketten benötigt werden. Darüber hinaus stellt es automatisch fest, ob ein oder zwei Laufwerke angeschlossen sind und verhält sich dann dementsprechend.
DISCKIT2 wird ganz normal mit

```
DISCKIT2 <ENTER>
```

geladen und gestartet.

Zunächst findet die automatische Abfrage statt, wieviele Laufwerke angeschlossen sind. Haben Sie nur das Laufwerk A in Betrieb, erscheint

```
One drive found
```

Ist dagegen ein zusätzliches Laufwerk B angeschlossen, lautet die Meldung

```
Two drives found
```

Im unteren Bildschirmteil erscheint nun folgendes Auswahlmenü:

```
     Copy            7          (Diskette kopieren)
     Format          4          (Diskette
     Verify          1          (Diskette
   Exit from program 0          (Diskette verlassen)
```

Zur Auswahl des gewünschten Menüpunktes müssen Sie die entsprechende Funktionstaste im Zifferntastenblock drücken. Diese Tasten sind nachfolgend mit <F0> bis <F9> gekennzeichnet.

Wenn Sie nun eine Diskette formatieren möchten, drücken Sie die Taste <F4>. Nehmen Sie jetzt die Systemdiskette aus dem Laufwerk und legen Sie stattdessen die zu formatierende Diskette ein. Achten Sie unbedingt darauf, daß Sie eine fabrikneue Diskette einlegen oder eine solche, auf deren Inhalt Sie unbedingt verzichten können. Durch die Formatierung gehen etwa vorhandene Daten auf der Diskette unwiderruflich verloren!

Sie sehen nun ein weiteres Menü auf dem Bildschirm, in dem Sie zur Art der Formatierung gefragt werden:

```
System format        9          (Systemformat)
Data format          6          (Datenformat)
Vendor format        3          (Vendorformat)
Exit menu                       (Menü verlassen)
```

Die hier angegebenen Formate kennen wir bereits aus dem FORMAT-Programm. DISCKIT2 hat lediglich den Nachteil, daß es nicht im IBM-Format formatieren kann.

Nachdem Sie die gewünschte Formatierungfart ausgewählt haben, er-scheint folgende Meldung auf dem Bildschirm (z.B. für das Systemformat):

```
Y       Format as System
Any other key to exit
menu
```

Drücken Sie jetzt Y, falls Sie im Systemformat formatieren möchten. Dies ist eine letzte Sicherheitsabfrage vor dem Formatieren. Falls Sie ein anderes Format gewählt haben, erscheint eine entsprechende ähnliche Meldung. Wenn Sie statt Y eine andere Taste drücken, findet keine Formatierung statt, und der Computer kehrt ins Menü zurück.

Falls zwei Laufwerke angeschlossen sind, erfolgt vor dem Formatieren noch zusätzlich die Abfrage, ob die zu formatierende Diskette in Lauf-werk A oder B liegt.

Kommen wir jetzt zum Kopieren einer Diskette (Taste <F7>). Falls die Zieldiskette noch nicht formatiert ist, wird dies automatisch während des Kopiervorgangs. nachgeholt.

Sie brauchen sich beim Kopieren nicht darum kümmern, ob die zu kopierende Diskette (Quelldiskette) im System-, Daten- oder Vendorformat formatiert ist, da dies der Computer automatisch erkennt. Falls Sie nur mit einem Laufwerk kopieren, erfolgt zunächst die Sicherheitsabfrage

```
Y      Copy
Any other key to exit menu
```

Wenn Sie diese mit Y (Ja) beantworten, kann der Kopiervorgang beginnen. Sie sollten zuvor jedoch Ihre Systemdiskette aus dem Laufwerk nehmen und stattdessen die Diskette einlegen, die kopiert werden soll. Nun erscheinen abwechselnd die Meldungen (bei einem Laufwerk)

```
Insert disc to WRITE          (Zieldiskette einlegen)
```



```
Press any key to continue      (Weiter - beliebige Taste drücken)
```

und

```
Insert disc to READ            (Quelldiskette enlegen)
Press any key to continue      (Weiter - beliebige Taste drücken)
```

Wenn Sie die Anweisungen befolgen und den Diskettenwechel immer richtig vornehmen, erscheint nach einigen Durchgängen die Meldung

```
Copy completed                 (Kopiervorgang beendet)
Remove disc                    (Diskette herausnehmen)
Press any key to continue      (Weiter - beliebige Taste drücken)
```

Damit ist der Kopiervorgang beendet.

Beim Kopieren erscheinen auf dem Bildschirm eigenartige Streifen. Dies ist jedoch kein Fehler und beruht darauf, daß DISCKIT2 beim Kopieren mit einem Laufwerk den Bildschirmspeicher zur Zwischenspeicherung der Sektoreninhalte mit benutzt. Dafür muß aber die Diskette nicht so häufig gewechselt werden wie bei DISCCOPY.

Falls sie mit zwei Laufwerken kopieren, müssen Sie Quell- und Zieldiskette nicht ständig wechseln. Zunächst erscheint das Zusatzmenü:

```
Write to A:      9          (auf Laufwerk A kopieren)
Write to B:      6          (auf Laufwerk B kopieren)
Exit menu        3          (Menü verlassen)
```

Sie legen jetzt Quell- und Zieldiskette in das richtige Laufwerk ein und drücken die entsprechende Taste. Es ist allerdings bequemer und schneller, von Laufwerk A auf Laufwerk B zu kopieren.

Der letzte Menüpunkt von DISCKIT2 ist Verify (Taste <F1>). Hierbei wird überprüft, ob eine Diskette richtig kopiert wurde. Quell- und Zieldiskette werden miteinander verglichen und alle eventuell auftretenden Fehler angezeigt.

Verify arbeitet ebenfalls wahlweise mit einem oder zwei Laufwerken. Beachten Sie alle Anweisungen, die auf dem Bildschirm erscheinen, um den Test durchzuführen.

Verify entspricht in seiner Funktion den Programmen DISCCHK bzw. CHKDISC.

## 2.7 Residente und nichtresidente Befehle

Unter residenten Befehlen versteht man all diejenigen Befehle, die fest ins CP/M- Betriebssystem eingebaut sind und nicht erst von der Diskette nachgeladen werden müssen.

Wir haben uns bereits mit den Befehlen DIR und STAT beschäftigt, um das Directory aufzulisten. Sicher erinnern Sie sich, daß ein Dienstprogramm DIR auf der Systemdiskette nicht vorhanden ist und Sie diesen Befehl jederzeit einsetzen konnten. DIR ist somit ein residenter Befehl, der jederzeit und unabhängig von der Systemdiskette erteilt werden kann.

STAT dagegen ist ein nichtresidenter Befehl. Wenn wir ihn verwenden, müssen wir zunächst darauf achten, daß die STAT.COM-Datei auf der zu untersuchenden Diskette oder beim Einsatz von zwei Laufwerken mindestens auf einem Laufwerk vorhanden ist. CP/M lädt diese Datei von Diskette und führt sie dann anschließend aus.

Hier eine Aufstellung der residenten und der wichtigsten nichtresidenten Befehle, die in jedem CP/M 2.2-System vorhanden sind, ohne Rücksicht darauf, ob sie bereits besprochen wurden oder nicht:



**Residente Befehle:**

| | |
|---|---|
| DIR | Listet Directory |
| ERA | Löscht Dateien |
| REN | Benennt Dateien um |
| SAVE | Speichert Dateien auf Diskette |
| TYPE | Listet ASCII-Datei |
| USER | Weist Benutzerbereich zu |

**Nichtresidente Befehle:**

| | |
|---|---|
| ASM | Assembler |
| DDT | Debugger |
| DUMP | Listet Maschinenprogramm |
| ED | Ruft den Texteditor auf |
| LOAD | Erzeugt COM-Datei aus HEX-Datei |
| MOVCPM | Anpassung des CP/M-Systems an die Speichergröße |
| PIP | Universelles Kommunikations- und Kopierprogramm |
| STAT | Fragt Systemzustand ab |
| SUBMIT | Dient zur Stapelverarbeitung |
| SYSGEN | Schreibt CP/M in Systemspuren |

## 2.8 Dateien umbenennen

Dateien können mit dem REN-Befehl einen neuen Namen erhalten. Dabei wird der alte Name lediglich durch den neuen im Directory ersetzt. Hier die allgemeine Schreibweise:

```
REN neuer Dateiname=alter Dateiname
```

Sowohl der neue als auch der alte Dateiname müssen jeweils einschließlich Extension angegeben werden. Dazu ein Beispiel:

```
REN NEUDAT.TXT=ALT DAT.TXT
```

Hierbei wird die Datei ALTDAT.TXT in NEUDAT.TXT umbenannt. Soll die Umbenennung auf Laufwerk B stattfinden, lautet der Befehl entsprechend

```
REN B:NEUDAT.TXT=ALTDAT.TXT
```

Achten Sie darauf, daß - außer direkt hinter REN - kein Leerzeichen in der

```
REN *.TXT=*.BAK
```

sind demgemäß nicht zugelassen, so daß Sie in einem solchen Fall jede Datai einzeln umbenennen müssen.

Ferner darf noch keine andere Datai unter dem neuen Namen auf der Diskette abgelegt sein. Ist dies doch der Fall, erscheint die Meldung:

```
FILE EXISTS
```

und die Umbenennung findet nicht statt.

## 2.9 Dateien löschen

Zum Löschen von Dateien dient der ERA-Befehl. Hinter ERA muß ein ein- oder mehrdeutiger Dateiname aufgeführt sein. Hier einige Beispiele:

```
ERA PIP.COM
```

löscht die Datei PIP.COM,

```
ERA B:PROBE.TXT
```

löscht die Datei PROBE.TXT auf Laufwerk B,

```
ERA *.COM
```

löscht sämtliche COM-Dateien,

```
ERA AB*.*
```

löscht sämtliche Dateien, deren Namen mit AB beginnen, und

```
ERA *.*
```

löscht sämtliche Dateien auf der Diskette. Da dieser Befehl bei falscher Anwendung fatale Auswirkungen haben kann, folgt vor dem eigentlichen Löschvorgang noch die Sicherheitsabfrage

```
ALL (Y/N)?
```

Wenn Sie nun Y eingeben, werden die Dateien gelöscht, bei N dagegen findet keine Löschung statt.



## 2.10 Fehlermeldungen

Beim CPC können unter CP/M zweierlei Fehlermeldungen auftreten. Betrachten wir zunächst die hardwarespezifischen, die vom Disc-Controller erzeugt werden und die auch unter BASIC erscheinen.

Versuchen Sie beispielsweise auf die Floppy zuzugreifen, wenn keine Diskette eingelegt ist, erscheint die Meldung

```
Drive A: disc missing
Retry, Ignore or
Cancel?
```

Wenn Sie jetzt eine Diskette einlegen und die Taste R (Retry) drücken, wird ein neuer Versuch unternommen, auf die Diskette zuzugreifen. Geben Sie dagegen I (Ignore) ein, versucht das BIOS, die Bearbeitung fortzusetzen, als wäre nichts geschehen, bei C (Cancel) dagegen wird die Bearbeitung ganz abgebrochen. In diesem Fall erscheint häufig ein Bdos-Error (s.u.), der nur durch einen Warmstart behoben werden kann.

Die zweite Gruppe von Fehlern wird durch das BDOS angezeigt, wobei folgende Fehlermeldungen auftreten können:

```
Bdos Err On A: Bad Sector (Diskettenfehler)

Bdos Err On A: Select     (angesprochenes Laufwerk nicht vorhanden)

Bdos Err On A: R/O        (Laufwerk ist schreibgeschützt)

Bdos Err On A: File R/O   (Datei ist schreibgeschützt)
```

Wenn einer dieser Fehler auftritt, hat man in der Regel nur die Möglichkeit, einen Warmstart auszulösen, indem man eine beliebige Taste drückt.

Am besten ist jedoch, wenn Sie dafür Vorsorge tragen, daß ein Bdos-Error niemals während eines ablaufenden Programms auftritt. Wenn Sie dann nämlich den Warmstart auslösen, wird das Programm endgültig abgebrochen. Anders ist es, wenn Sie, wie im ersten Beispiel, vergessen haben, die Diskette einzulegen und Sie dies lediglich nachholen und die R-Taste drücken müssen.

Trotz allem ist aber die Anfertigung von Sicherheitskopien immer noch das beste Mittel, um größeren Schaden abzuwenden. Wenn Sie beispielsweise eine Textdatei bearbeiten, sollten Sie sie von Zeit zu Zeit auf eine andere Diskette kopieren. Dann nämlich hält sich im Falle eines Fehlers der Schaden in Grenzen, selbst wenn das gesamte System ausfällt oder die Diskette zerstört wird.

Falls Sie mit zwei Laufwerken arbeiten, sollten Sie die Programmdiskette sicherheitshalber schreibschützen und immer in Laufwerk A einlegen. Gemeint ist hier der Schreibschutz an der Diskette, indem Sie die kleine rote Klappe öffnen (s. Kapitel 1). In Laufwerk B kommt dann die Diskette mit den Dateien, die Sie bearbeiten.

## 2.11 Das Arbeiten mit gekaufter Software

Wenn Sie fertige CP/M-Software kaufen, die bereits für den CPC angepaßt ist, ist sie in der Regel sofort einsatzbereit. Ist dies nicht der Fall, liegt gewöhnlich eine Beschreibung bei, was Sie noch tun müssen. Meistens handelt es sich dabei um Dinge wie Auswahl der Bildschirmfarben, Änderung der Tastaturbelegung oder die Einbeziehung von Sonderzeichen. Zu diesem Zweck befindet sich normalerweise ein zusätzliches Programm auf der Diskette, das einen Namen wie CONFIG.COM o.ä. trägt. Wenn Sie dieses Programm ausführen, benötigen Sie keine besonderen Kenntnisse, denn sämtliche Instruktionen werden meist ausführlich auf dem Bildschirm oder im beiliegenden Handbuch erklärt.

Bevor Sie nun mit der Software arbeiten, empfiehlt es sich, eine Arbeitskopie herzustellen, wie oben beschrieben, und die Orginaldiskette an einem sicheren Ort aufzubewahren. Fertige Software wird meist auf Disketten geliefert, die im Vendor-Format (s.o.) formatiert sind und somit das CP/M-Betreiebssystem nicht enthalten.

Zur Herstellung der Arbeitskopie gibt es zwei Möglichkeiten: Entweder Sie formatieren eine Diskette im Systemformat und übertragen dann da-rauf die Dateien der Orginaldiskette mit Hilfe von FILECOPY oder PIP. Sie können aber auch die gesamte Orginaldiskette komplett kopieren, müssen dann aber noch gesondert das CP/M-Betriebssystem und den BOOT-Sektor auftragen. Zu diesem Zweck befinden sich auf Ihrer Systemdiskette zwei Programme, nämlich SYSGEN und BOOTGEN.

Zunächst übertragen wir mit SYSGEN das CP/M-Betriebssystem von einer anderen Diskette, die es bereits enthält, z.B. von der Systemdiskette. Dabei laden Sie das Programm mit

```
SYSGEN
```

von der Systemdiskette, worauf folgende Meldungen erscheinen:

```
Please insert SOURCE disc into drive A and press any key:
```



```
(Bitte Quelldiskette in A einlegen und beliebige Taste drücken:)
```

Als Quelldiskette verwenden wir gleich die Systemdiskette und drücken eine beliebige Taste. SYSGEN liest jetzt die Systemspuren in den Arbeitsspeicher. Daraufhin folgt

```
Please insert DESTINATION disc into drive A and press any key: (
Bitte Zieldiskette in A einlegen und beliebige Taste
drücken:)
```

Jetzt nehmen Sie die Systemdiskette aus dem Laufwerk heraus und legen statt dessen die Arbeitsdiskette mit Ihrer Software ein. Sie drücken eine beliebige Taste und das CP/M-Betriebssystem wird in die Systemspuren geschrieben.

Dasselbe wiederholen Sie jetzt mit dem Programm BOOTGEN, das den BOOT-Sektor auf die Diskette schreibt. Die Vorgehensweise ist hier genauso wie bei SYSGEN.

Der BOOT-Sektor ist für das ordnungsgemäße Laden von CP/M zuständig. Wenn Sie nämlich

```
1CPM
```

eingeben, wird zunächst der BOOT-Sektor geladen, der dann wieder das CP/M-Betriebssystem nachlädt.

# 3 STAT

STAT ist ein CP/M-Dienstprogramm, mit dessen Hilfe man wichtige Informationen über das verwendete CP/M-System, Disketten oder Dateien erhält. Darüber hinaus dient es auch zum Setzen und Aufheben des Schreibschutzes von Dateien und zur Bestimmung oder Änderung der Gerätezuordnungen.

In Kapitel 2 haben wir bereits einmal mit STAT gearbeitet. Dort listeten wir mit Hilfe von STAT das Directory einer Diskette in einer etwas ausführlicheren Form auf, als es mit dem DIR-Befehl möglich ist. Strenggenommen haben wir aber nicht das Directory, sondern den Status oder Zustand sämtlicher Dateien auf der Systemdiskette aufgezeigt. Dabei handelte sich jedoch nur um einen Anwendungsfall von mehreren, mit denen wir uns nachfolgend näher befassen werden.

Da STAT ein nichtresidenter Befehl ist und aus einer COM-Datei besteht, muß er immer auf Diskette in einem Laufwerk abgelegt sein. Wenn Sie mit zwei Laufwerken arbeiten, ist es am einfachsten, mit STAT zu arbeiten. Legen Sie deshalb die Systemdiskette mit sämtlichen Dienstprogrammen in Laufwerk A und die Diskette, die zu untersuchen ist, in Laufwerk B.

Falls Sie nur ein Laufwerk zur Verfügung haben, ist es unerläßlich, STAT auf die zu untersuchende Diskette zu kopieren. Dazu verwenden Sie am besten das Programm FILECOPY, wie in Kapitel 2 beschrieben.

## 3.1 Freie Diskettenkapazität

Die einfachste Möglichkeit, STAT einzusetzen, ist die Ermittlung des freien Speicherplatzes auf der Diskette. Dazu rufen Sie einfach STAT ohne jeglichen Zusatz auf:

STAT <ENTER>

Dabei erscheint eine Meldung, die angibt, wieviel freier Speicherplatz noch auf der (den) Diskette(n) vorhanden ist und ob ein Schreibschutz für das Laufwerk angebracht ist oder nicht. Gemeint ist hier allerdings ein Schreibschutz, den das CP/M-Betriebssystem intern setzt, und der vor je-dem Schreibzugriff abgefragt wird. Er darf keinesfalls mit dem Schreibschutz verwechselt werden, der durch Betätigen des Schiebers bzw. der roten Klappe am Diskettengehäuse erreicht wird.

Hier ein Beispiel, in dem die Meldung für zwei Laufwerke ausgegeben wird:

```
A: R/W, Space: 44k
B: R/O, Space: 51k
```

Wir sehen also, daß Laufwerk A und B gleichzeitig abgefragt werden. Da CP/M theoretisch 16 verschiedene Laufwerke verwalten kann, werden mit diesem Befehl alle angeschlossenen Laufwerke überprüft. Der CPC ist allerdings nur für maximal zwei Laufwerke ausgelegt.

Im vorliegenden Beispiel ist Laufwerk A für Schreib- und Lesevorgänge (R/W) zugelassen und erhält noch 44K an freiem Speicherplatz, wodurch der Belegungsgrad der Diskette wiedergegeben wird.

Laufwerk B dagegen enthält einen Schreibschutz (R/O) und kann noch 51K an Dateien aufnehmen.

Wenn Sie nur ein Laufwerk angeschlossen haben, erscheint auch nur der Zustand von Laufwerk A, im vorliegenden Beispiel also

```
A: R/W, Space: 44k
```

Wenn Sie hinter STAT eine Laufwerksbezeichnung angeben, erhalten Sie allerdings keine Auskunft über den angebrachten Schreibschutz. So ergibt beispielsweise

```
STAT A:
```

die Meldung

```
Bytes Remaining On A: 44k
```

und bei

```
STAT B:
```

erscheint die Meldung (nur bei zwei Laufwerken!)

```
Bytes Remaining On B: 51k
```



## 3.2 Zustand von Dateien

Ebenso wie für Disketten können wir auch für jede einzelne Datei deren Zustand ermitteln. Hier wiederholen wir nun das, was wir in Kapitel 2 bereits kennengelernt haben, dort allerdings im Zusammenhang mit der Auflistung des Directories.

Zur Ermittlung des Dateizustandes müssen wir hinter STAT noch einen Dateinamen angeben. Wenn uns beispielsweise die Datei LAGER.DAT interessiert, lautet die Befehlszeile

```
STAT LAGER.DAT
```

Daraufhin erscheint die Meldung

```
Recs    Bytes   Ext Acc
147     19k     2 R/W A:LAGER
DAT Bytes Remaining On A: 78k
```

Die Datei LAGER.DAT ist nicht schreibgeschützt (R/W) und umfaßt 147 Records. Diese belegen insgesamt 19k auf der Diskette und benötigen 2 Einträge im Directory (Extents). Außerdem stehen 78K an freiem Speicherplatz zur Verfügung.

In Kapitel 2 hatten wir uns bereits mit den Speichereinheiten auf der Diskette näher befaßt. Fassen wir noch einmal zusammen:

CP/M verwaltet, unabhängig von der physikalischen Diskettenstruktur, sämtliche Dateien in Records zu je 128 Bytes. Die interne Dateiverwaltung ist aber so ausgelegt, daß für jede Datei eine bestimmte Anzahl von Blöcken (1024 Bytes) reserviert wird. Da ein Eintrag im Directory aber nur 16 Blöcke erfassen kann, ist für jede weiteren angefangenen 16K ein weiterer Eintrag ( Extent) erforderlich.

Wir sagten bereits, daß zur Abfrage des Dateizustandes der betreffende Dateiname anzugeben ist. Dabei muß es sich aber keinesfalls nur um einen eindeutigen Namen handeln, wie es im obigen Beispiel der Fall ist. Zulässig sind auch alle mehrdeutigen Dateinamen, so wie wir sie im Zusammenhang mit dem DIR-Befehl in Kapitel 2 kennengelernt haben. Hier einige Beispiele:

```
STAT *.COM
```

ermittelt den Zustand sämtlicher COM-Dateien,

```
STAT B:LA*.DAT
```

den Zustand sämtlicher Dateien in Laufwerk B, deren Name mit LA beginnt und die Extension DAT enthält und

STAT *.*

listet den Zustand sämtlicher Dateien im Bezugslaufwerk. Wenn wir das letzte Beispiel mit der Systemdiskette, die zum CPC geliefert wird, ausprobieren, erhalten wir:

```
Recs  Bytes  Ext Acc
   2    1k   1 R/W A:AMSDOS.COM
  64    8k   1 R/W A:ASM.COM
  10    2k   1 R/W A:BOOTGEN.COM
  19    3k   1 R/W A:CHKDISC.COM
  15    2k   1 R/W A:CLOAD.COM
  21    3k   1 R/W A:COPYDISC.COM
  14    2k   1 R/W A:CSAVE.COM
  38    5k   1 R/W A:DDT.COM
  19    3k   1 R/W A:DISCCHK.COM
  21    3k   1 R/WA:DISCCOPY.COM
  33    5k   1 R/WA:DUMP.AOM
   4    1k   1 R/WA:DUMP.COM
  52    7k   1 R/WA:ED.COM
   9    2k   1 R/WA:EX1.BAS
   3    1k   1 R/WA:EX2.BAS
  22    3k   1 R/WA:FILECOPY.COM
  21    3k   1 R/WA:FORMAT.COM
  14    2k   1 R/WA:LOAD.COM
  76   10k   1 R/WA:MOVCPM.COM
  58    8k   1 R/WA:PIP.COM
 208   26k   2 R/WA:ROINTIME.DEM
  61    8k   1 R/WA:SETUP.COM
  41    6k   1 R/WA:STAT.COM
  10    2k   1 R/WA:SUBMIT.COM
  12    2k   1 R/WA:SYSGEN.COM
   6    1k   1 R/WA:XSUB.COM
Bytes Remaining On A: 50k
```

Drücken wir vorher noch <CTRL-P>, schalten wir den Drucker hinzu, so daß wir die gesamte Auflistung auch ausdrucken können.

Wurde eine Datei oder Dateigruppe mit mehrdeutigem Namen nicht auf dem angesprochenen Laufwerk gefunden, erscheint die Meldung

```
File Not Found
```

## 3.3 Laufwerkseigenschaften

Mit einer weiteren STAT-Variante können wir nähere Enzelheiten über ein Diskettenlaufwerk in Erfahrung bringen. Dies erreichen wir mit



STAT DSK:

für das Bezugslaufwerk oder mit

STAT B:DSK:

für Laufwerk B.

Mit diesem Befehl erhalten wir für ein CPC-Laufwerk folgende Angaben:

```
   A:   Drive Characteristics
1368:   128 Byte Record Capacity
 171:   Kilobyte Drive Capacity
  64:   32 Byte Directory Entries
  64:   Checked Directory Entries
 128:   Records/Extent
   8:   Records/Block
  36:   Sectors/Track
   2:   Reserved Tracks
```

Beginnen wir von oben nach unten: In unserem Beispiel wurden die Eigenschaften von Laufwerk A aufgelistet, welches am Anfang der ersten Zeile angegeben ist. Für Laufwerk B würden wir aber die gleichen Angaben erhalten, sofern es sich dabei ebenfalls um ein 3-Zoll-CPC-Laufwerk handelt.

Die zweite Zeile gibt an, daß das Laufwerk eine Kapazität von 1368 Records zu je 128 Bytes umfaßt. Dabei sind allerdings die Systemspuren nicht mit enthalten. Insgesamt können wir also

1368 * 128 = 175104 Bytes

auf der Diskette mit Dateien belegen, was

175104 / 1024 = 171K (Kbytes)

entspricht. Somit sind wir auch schon bei der dritten Zeile, die diesen Wert angibt.

Die nächsten beiden Zeilen machen Angaben über das Directory. Wir erfahren, daß ein Directoryeintrag intern 32 Bytes beansprucht und wir maximal 64 Einträge vornehmen können. Sämtliche 64 Einträge können wir auch mit einem Schreibschutz versehen (R/O), was durch den Zusatz " Checked" angegeben wird.

Als weiteres erfahren wir, daß ein Verzeichniseintrag (Extent) maximal 128 Records verwalten kann. Wenn wir dies nachrechnen, erhalten wir

128 * 128 = 16384 Bytes oder 16K

pro Extent, wie wir bereits oben gesehen haben.

Darüber hinaus befinden sich 8 Records in einem Block (1024 Bytes) und *36* Sektoren auf einer Spur (Track). Die letzte Angabe ist leider nicht ganz richtig und müßte eigentlich Records/Track lauten. Hierbei handelt es sich offensichtlich noch um ein Überbleibsel aus der Anfangszeit von CP/M, als Records und Sektoren auf einem 8-Zoll-IBM-Laufwerk noch identisch waren. Das CPC-Laufwerk enthält nämlich 9 Sektoren pro Spur, von denen jeder 512 Bytes an Informationen aufnimmt.

Schließlich gibt die letzte Zeile an, daß zwei Spuren reserviert sind. Dies betrifft die beiden äußeren Spuren 0 und 1, die den Boot-Sektor und das CP/M-Betriebssystem aufnehmen und deshalb nicht zum Speichern von Dateien zur Verfügung stehen.

## 3.4 Gerätezuordnung

In diesem Abschnitt lernen wir einen STAT-Befehl kennen, der die Konfiguration des IOBYTEs wiedergibt. Das IOBYTE ist in Speicheradresse 3 abgelegt und gibt die Zuordnung der Ein- und Ausgabekanäle an.

Wenn Sie jetzt

STAT DEV:

eingeben, so erhalten Sie folgende Angaben:

```
CON:  s  CRT:
RDR:  is  TTY:
PUN:  s  TTY:
LST: s LPT:
```

Diese Angaben beziehen sich auf die Ein-/Ausgabekanäle, denen jeweils ein bestimmtes Gerät zugeordnet ist. CP/M besitzt insgesamt vier Kanäle, die wir uns einmal näher anschauen wollen:

**CON:**

CON: steht für Konsole oder Terminal. Dem Konsolenkanal ist in der Regel ein Bildschirmgerät mit Tastatur zugeordnet, das hier mit CRT: (Cathode Ray Tube) angegeben wird. Dies mag Ihnen vielleicht etwas ungewöhnlich vorkommen, da Sie auf Ihrem CPC ohnehin eine Tastatur und einen Bildschirm haben. In der Entstehungszeit von CP/M war das



jedoch gar nicht selbstverständlich und man mußte sämtliche Teileinheiten eines Computers erst einmal zusammensetzen, bevor man mit ihm arbeiten konnte.

Theoretisch kann, man dem Konsolenkanal außer CRT: noch andere Geräte zuordnen, was aber auf dem CPC nicht ratsam ist. Das IOBYTE ist nämlich mit folgenden Angaben konfigurierbar:

| | |
|---|---|
| CRT: | Bildschirm/Tastatur |
| TTY: | Fernschreiber |
| BAT: | Stapelverarbeitung (heute ohne Bedeutung) |
| UC1: | Vom Benutzer selbst definiertes Terminal |

**RDR:**

RDR steht für "Reader" oder Lochstreifenleser, der allerdings heute kaum noch verwendet wird. Hier sind theoretisch folgende Zuordnungen möglich:

| | |
|---|---|
| TTY: | Fernschreiber |
| PTR: | besonders schneller Lochstreifenleser |
| UR1: | Vom Benutzer definiertes Lesegerät |
| UR2: | Vom Benutzer definiertes Lesegerät |

**PUN:**

Dies ist der Kanal für den Lochstreifenstanzer (Puncher), ein ebenfalls heute antiquiertes Gerät, mit folgenden möglichen Zuordnungen:

| | |
|---|---|
| TTY: | Fernschreiber |
| PTP: | Besonders schnelles Stanzgerät |
| UP1: | Vom Benutzer definiertes Ausgabegerät |
| UP2: | Vom Benutzer definiertes Ausgabegerät |

**LST:**

Über diesen Kanal ist normalerweise der Drucker angeschlossen, jedoch können ihm folgende Geräte zugeordnet werden:

| | |
|---|---|
| LPT: | Drucker (Line Printer) |
| TTY: | Fernschreiber |
| CRT: | Bildschirm |
| UL1: | Vom Benutzer definiertes Ausgabegerät |

Die angegebenen Gerätezuordnungen erscheinen Ihnen sicher etwas abstrakt, aber Sie brauchen sich mit den meisten sowieso nicht näher zu

befassen. Außerdem sind diese Zuordnungen nur wirksam, wenn sich im BIOS eine entsprechende Routine zur Ansteuerung dieser Geräte befindet.

Für Sie als CPC-Anwender wäre es allerdings durchaus denkbar, daß Sie an Ihren Computer über eine zusätzliche serielle Schnittstelle ein Modem oder einen Akustikkoppler zur Datenfernübertragung anschließen. Hierzu könnten Sie den RDR:- und PUN:-Kanal zweckentfremden, wenn sich gleichzeitig im BIOS die entsprechenden Treiberroutinen befinden. Dabei kann durchaus die Zuordnung TTY (Fernschreiber) beibehalten werden.

Ein weiterer Anwendungsfall wäre gegeben, wenn Sie die Druckerausgabe, die standardmäßig mit LPT: über die Centronics-Schnittstelle statt-findet, auf die serielle Schnittstelle umlenken, um entweder einen seriellen Drucker zu betreiben oder aber anstelle des Druckers ein Modem einzusetzen. Hier bietet STAT nun einen Befehl, mit dem Sie die Schnittstelle anstelle des Druckers über den LST:-Kanal ansteuern können:

```
STAT LST:=TTY:
```

Wenn Sie später wieder den Drucker zuordnen möchten, lautet der Befehl

```
STAT LST:=LPT:
```

Beachten Sie aber, daß TTY normalerweise keine Wirkung zeigt, es sei denn, Sie schließen über den Expansionsport ein Gerät an, das diese Zuordnung benutzt.

Wenn Sie einmal ohne Drucker arbeiten müssen, kann es leicht vorkommen, daß Sie aus Versehen <CTRL—P> drücken und sich infolgedessen der Rechner aufhängt, da kein Drucker angeschlossen ist. In diesem Fall ist es eine gute Schutzmaßnahme, wenn Sie dem LST:-Kanal ein nicht verwendetes Gerät wie z.B. TTY: zuordnen, denn dann geschieht infolge von <CTRL—P> offenbar gar nichts.

## 3.5 Schreibschutz für Laufwerk

Mit folgendem Befehl können Sie ein Laufwerk vor Schreibzugriffen schützen, wodurch es mit R/O gekennzeichnet ist (s. Absatz 3.1):

```
STAT A:=R/0
```

bzw.

```
STAT B:=R/0
```



Die Laufwerksbezeichnung ist hier unbedingt mit anzugeben. Es genügt also nicht, sich auf das Bezugslaufwerk zu stützen.

Dieser Befehl ändert nichts auf der Diskette und teilt CP/M lediglich mit, welches Laufwerk zu schützen ist. Der Schreibschutz kann nur durch einen Warmstart mit CTRL-C wieder aufgehoben werden.

## 3.6 Dateiattribute

Neben der Möglichkeit, ein Laufwerk schreibzuschützen, ist dies auch für einzelne Dateien möglich. Außerdem können Dateien auch als Systemdateien bestimmt werden, so daß sie mit dem DIR-Befehl nicht mehr aufgelistet werden können. Ansonsten sind sie aber ganz normal aufzurufen bzw. auszuführen.

Beides ist durch das Setzen sogenannter Dateiattribute zu erreichen, wobei im Directory ein entsprechender Vermerk angebracht wird.
Hier einige Beispiele:

```
STAT B:MITGLIED.DAT $R/O
```

versieht in Laufwerk B die Datei MITGLIED.DAT mit einem Schreibschutz. Infolge von

```
STAT B:MITGLIED.DAT $R/W
```

wird der Schreibschutz wieder aufgehoben.

Soll ein Dateiname nicht mit DIR auflistbar sein, geben Sie als Attribut $SYS und zur Aufhebung des Listschutzes $DIR vor. So verhindert z.B.

```
STAT TEST.COM $SYS
```

daß die Datei TEST.COM aufgelistet werden kann, wogegen

```
STAT TEST.COM $DIR
```

den Listschutz wieder aufhebt.

Diese Maßnahme hat allerdings einen Pferdefuß, denn Systemdateien können dennoch gelistet werden, und zwar in unserem Beispiel mit

```
STAT TEST.COM
```

Dabei erschient der listgeschützte Dateiname lediglich in Klammern:

```
Recs   Bytes Ext Acc
10       2K    1 R/W A:(TEST.
COM) Bytes Remaining On A: 67k
```

Auch bei der Verwendung von Dateiattributen sind mehrdeutige Dateinamen zulässig, so daß auch ganze Dateigruppen auf einmal mit einem Schreib- bzw. Listschutz versehen werden können.

## 3.7 Benutzerbereiche ermitteln

Unter CP/M kann man mit maximal 16 Benutzerbereichen arbeiten. Mit dem Befehl

USER 4

wird beispielsweise der Benutzerbereich 4 ausgewählt, so daß nur die Dateien, die darin abgelegt sind, aufgerufen bzw. bearbeitet werden können. Mit

STAT USR:

werden die gegenwärtige Benutzernummer sowie die Nummern all derjenigen Benutzer aufgezeigt, die Dateien im Bezugslaufwerk abgelegt haben, wie im folgenden Beispiel:

```
Active User: 4
Active Files: 1 4 7 12
```

Standardmäßig ist die USER-Nr. 0 voreingestellt. Der USER-Befehl ist auf dem CPC wenig sinnvoll und eignet sich in erster Linie nur für Mehrplatzsysteme, die z.B. unter MP/M arbeiten, in denen jeder Benutzer seinen eigenen Bereich erhält.

## 3.8 Zusammenfassung der STAT-Befehle

Mit dem Befehl

STAT VAL:



erhalten Sie eine Übersicht sämtlicher STAT-Befehle, die folgendermaßen auf dem Bildschirm erscheinen:

```
Temp R/O Disk: d:=R/O
Set Indicator: d:filename.typ $R/O $R/W SSYS $DIR
Disk Status : DSK: d:
DSK: User Status : USR:
Iobyte Assign:
CON:  =  TTY:  CRT:  BAT:
UC1:  RDR:  =  TTY:  PTR:
UR1:  UR2:  PUN:  =  TTY:
PTP:  UP1:  UP2:  LST:  =
TTY: CRT: LPT: UL1:
```

Dabei steht "d:" für die Laufwerksbezeichnung. Sämtliche hierin aufgeführten Befehle haben wir in diesem Kapitel bereits kennengelernt.

# 4 Der Editor

## 4.1 Was ist ein Editor?

Den Begriff Editor haben wir bereits im ersten Kapitel kennengelernt. Allgemein handelt es sich dabei um ein internes Maschinenprogramm, das dem Benutzer gestattet, Texte in den Computer einzugeben oder vorhandene Texte zu korrigieren.

Was ist in diesem Zusammenhang mit einem Text gemeint? Sicher werden Sie jetzt antworten, daß z.B. die Zeilen, die Sie in diesem Buch lesen, einen Text darstellen. Damit haben Sie vollkommen recht, aber lassen Sie uns den Begriff Text einmal etwas genauer definieren.

Ein Text ist eine beliebige Aneinanderreihung von Text- oder ASCII-Zeichen, im Gegensatz zu jedem anderen Code, wie beispielsweise dem internen BASIC- oder Maschinencode. Dem Computer ist es letztlich egal, ob Sie in seinem Speicher Textzeichen oder etwas anderes ablegen, es kommt nur darauf an, nach welcher Definition er die gespeicherten Informationen handhabt. So gibt es den Z80-Maschinencode, der ganz bestimmten Regeln unterliegt, damit ihn die CPU richtig abarbeitet. In einem Maschinenprogramm ist also festgelegt oder die Vereinbarung getroffen, daß es sich hierbei um Maschineninstruktionen handelt. Entsprechendes gilt auch für BASIC-Programme, wie sie intern im Speicher abgelegt sind. Hier wurde nämlich die Vereinbarung getroffen, daß es sich um einen Code handelt, den der BASIC-Interpreter "versteht" und abarbeiten kann.

Ganz gleich, ob Sie nun ein BASIC- oder Maschinenprogramm schreiben oder nur normale Texte eingeben, Sie benötigen in jedem Fall einen Editor, mit dem Sie die entsprechenden Informationen in einer allgemein lesbaren Form eingeben bzw. bearbeiten können. Bei einem Text, wie im vorliegenden Buch, ist dies noch am einfachsten zu verstehen, denn für jedes Textzeichen gibt es einen äqivalenten ASCII-Code, der im Anhang in einer Tabelle vollständig aufgeführt ist. So entspricht z.B. das Zeichen

"A" dem ASCII-Code *65*, das Zeichen "B" dem Code *66 usw*. Ähnliches gilt auch für die Kleinbuchstaben, die mit dem Wert 97 beginnen. Auch den Ziffern und Satzzeichen, wie Komma oder Punkt, ist ein ASCII-Wert zugeordnet.

Wenn Sie z.B. das Wort "Hallo!" mit dem Computer erfassen und Sie sich die entsprechenden Speicherzellen ansehen, finden Sie folgenden ASCII-Code vor:

| | | | | | | |
|---|---|---|---|---|---|---|
| 48 | 61 | 6C | 6C | 6F | 21 | (Hexadezimale Schreibweise) |
| 72 | 97 | 108 | 108 | 111 | 33 | (Dezimale Schreibweise) |
| H | a | 1 | 1 | o | ! | (Entsprechende Textzeichen) |

Ob Sie nun die hexadezimale oder die dezimale Darstellung erhalten, hängt davon ab, auf welche Weise Sie die Speicherzellen auflisten. Wenn Sie dies von BASIC aus mit dem PEEK-Befehl vornehmen, erhalten Sie in der Regel dezimale Zahlenwerte, während ein Debugger meist hexadezimal arbeitet. Ein Debugger ist übrigens ein Hilfsprogramm, zum Auflisten und Abändern von Speicherinhalten. Wir werden uns damit noch ausführlich in Kapitel 7 befassen.

Wenn nun der Computer auf diese Zeichen stößt und "weiß", daß er einen Text ausgeben soll, erscheint entsprechend das Wort "Hallo!". Weiß er es aber nicht, und wurde fälschlicherweise die Vereinbarung getroffen, daß es sich bei diesen Informationen um einen Maschinencode handelt, so hat dies wahrscheinlich verheerende Folgen.

Ein Editor erfaßt also immer Textzeichen und legt sie zunächst in Form von ASCII-Codes im Speicher ab. Natürlich stellt sich dann die Frage, was mit diesen Codes weiter geschieht. Der BASIC-Editor z.B. wandelt die Textzeichen in den internen BASIC-Code um, der dann vom Interpreter abgearbeitet werden kann.

Selbst wenn wir ein Maschinenprogramm schreiben, kann dies nur auf Umwegen über einen Editor, gleich welcher Art, geschehen, da wir den Maschinencode in lesbarer Form editieren müssen, bevor er in die Speicherzellen eingeschrieben wird.

Wollen wir nun umgekehrt das Wort "Hallo!" nicht im Klartext, sondern in Form von einzelnen ASCII-Codes mit einem Debugger in den Speicher schreiben, müssen wir selbst diese Hexcodes zuerst editieren. Wenn Sie anstelle des Zeichens "H" den ASCII-Code 48 eingeben, handelt es sich dabei immer noch um die beiden lesbaren Zeichen "4" und "8", die ebenfalls einen lesbaren ASCII-Code besitzen, der erst in das entsprechende Bitmuster umgewandelt werden muß.



Kurzum, was wir auch immer in lesbarer Form in den Computer ein-oder darauf ausgeben, wir benötigen auf jeden Fall eine Art Editor, der die Informationen textmäßig erfaßt bzw. wiedergibt. So spielt es letztlich keine Rolle, was für einen Text Sie mit dem Editor schreiben. Die ASCII-Codes werden - von einem reinen Schrifttext einmal abgesehen - erst nach dem Editieren weiterverarbeitet bzw. in einen anderen Code umgewandelt.

Auch CP/M enthält einen Editor, mit dem Sie beliebige Texte erfassen können; wir werden uns nachfolgend damit beschäftigen. Allerdings ist dieser Editor, im Gegensatz zu professionellen Textverarbeitungssystemen, nicht gerade bedienerfreundlich und stammt noch aus den frühesten Zeiten von CP/M. Sollten Sie aber ein richtiges Textverarbeitungsprogramm wie etwa WORDSTAR zur Verfügung haben, können Sie damit sämtliche Aufgaben, die sonst der CP/M-Editor übernimmt, wesentlich komfortabler ausführen. Auch WORDSTAR ist letztlich nur ein Texteditor, der Texte erfaßt und in ASCII-Code umwandelt. Wenn Sie dieses Programm allerdings nicht zur reinen Texterfassung benutzen, sondern beispielsweise zur Assemblerprogrammierung, sollten Sie auf jeden Fall besondere Steuercodes vermeiden, die z.B. zum Randausgleich, Fettdruck usw. dienen und nur von dem jeweiligen Textverarbeitungssystem richtig interpretiert werden.

## 4.2 Das Dienstprogramm ED

ED ist eine COM-Datei und enthält den CP/M-Texteditor. ED muß immer im Zusammenhang mit dem Dateinamen aufgerufen werden, unter dem die zu bearbeitende Datei auf Diskette abgelegt ist bzw. abgelegt werden soll.

Falls Sie nur mit einem Laufwerk arbeiten, sollten Sie als erstes eine Diskette im Systemformat formatieren und darauf die Datei ED.COM kopieren, beispielsweise mit FILECOPY. Die gleiche Diskette enthält später auch die Texte.

Bei zwei Laufwerken lassen Sie am besten die Systemdiskette wieder in Laufwerk A. In Laufwerk B legen Sie dann die Diskette, auf der die Texte abgelegt werden. Es ist völlig ausreichend, wenn diese Diskette nur im Datenformat formatiert ist, denn es wird niemals auf die Systemspuren in Laufwerk B zugegeriffen. Daurch steht Ihnen auch etwas mehr Platz für Dateien zur Verfügung.

Wir wissen bereits, daß ED nicht gerade den komfortabelsten Texteditor darstellt und nur zur Erfassung kleinerer Texte zu empfehlen ist. Nachfolgend werden wir anhand von Beispielen schrittweise lernen, mit ED umzugehen.

### 4.2.1 **Textdatei anlegen**

Wir wollen jetzt eine Textdatei mit nur wenigen Zeilen erstellen und unter dem Namen PROBE.TXT auf Diskette ablegen. Dazu geben wir bei einem Laufwerk

ED PROBE.TXT und

bei zwei Laufwerken ED

B:PROBE.TXT

ein. Da auf der Diskette noch keine Datei unter dem Namen PROBE.TXT vorhanden ist, erscheint die Meldung:

```
NEW FILE
```

und der Cursor steht hinter dem Sternchen. Dies zeigt an, daß der Editor bereit ist, Befehle entgegenzunehmen.

Daraufhin geben wir den Buchstaben i ein und drücken die ENTER-Taste. ED befindet sich jetzt im Insert- oder Einfügemodus, und auf dem Bildschirm erscheint:

```
NEW FILE
 *i 1:
```

Bei den meisten Befehlen spielt es keine Rolle, ob sie in Groß- oder Kleinbuchstaben eingegeben werden. Beim Einfügen mit i müssen wir aber aufpassen, denn ein kleines i läßt die Eingabe von Groß- und Kleinschrift zu, während ein großes I sämtliche Zeichen in Großschrift umwandelt. Ähnliches gilt auch für den S-(Ersetze-)Befehl (s.u.).

Die Ziffer 1 mit dem Doppelpunkt steht bereits für die erste Textzeile, die wir nun in Groß- und Kleinschrift eingeben können, da wir i klein gewählt haben. In diese Zeile schreiben wir:

```
Dies ist Zeile 1.
```



Falls Sie sich vertippen sollten, können Sie durch wiederholtes Drücken von <CTRL¯H> bis zum fehlerhaften Zeichen zurückgehen und ab dieser Stelle den Text neu eingeben. Soll die gesamte Zeile gelöscht werden, können Sie dazu auch <CTRL¯X> eingeben. Verwenden Sie aber auf gar keinen Fall die DEL- oder CLR-Taste, denn diese Tasten sorgen hier für unliebsame Überraschungen.

Wenn Sie die erste Zeile richtig eingegeben haben, drücken Sie die ENTER-Taste, worauf die zweite Textzeile mit der Ziffer 2 erscheint. Wir geben daraufhin den Text:

```
Dies ist Zeile 2.
```

wieder auf die gleiche Weise wie zuvor ein. Genauso verfahren wir auch mit den Zeilen 3, 4 und 5, wobei der Text jeweils die Nummer der Zeile angeben soll. Haben wir die fünfte Zeile eingegeben, steht der Cursor auf der sechsten. Hier brechen wir die Eingabe ab und geben <CTRL—Z> ein. Dieses Zeichen hat den ASCII-Code 26 bzw. IAH und spielt in der Textverarbeitung eine wichtige Rolle, denn es kennzeichnet das Textende.

Unter der sechsten Zeile erscheint ein Sternchen, so daß wir einen neuen Befehl eingeben können. Durch das Drücken der E-Taste speichern wir den Text auf Diskette ab, worauf nach Beendigung das Anforderungszeichen A> erscheint. Insgesamt muß dann auf dem Bildschirm folgendes stehen:

```
NEW FILE

:  *i                    (Einfügemodus setzen)
1:  Dies  ist  Zeile  1.
2:  Dies  ist  Zeile  2.
3:  Dies  ist  Zeile  3.
4:  Dies  ist  Zeile  4.
5:  Dies  ist  Zeile  5.
6:                       (Hier haben wir <CTRL—Z> eingegeben)
   *E                    (Datei sichern und ED verlassen)
```

Jetzt wollen wir einmal überprüfen, ob der Text auch richtig auf Diskette geschrieben ist. Dazu verwenden wir den residenten Befehl TYPE, der eine ASCII-Datei liest und listet.

Achtung! Verwenden Sie TYPE immer nur für Text- oder ASCII-Dateien. Enthält nämlich die Datei einen anderen als den ASCII-Code, kann es zu merkwürdigen Bildschirmreaktionen kommen, wenn nichtdarstellbare Zeichen ausgegeben werden.

Geben Sie nun

```
TYPE PROBE.TXT
```

bei einem Laufwerk und

```
TYPE B:PROBE.TXT
```

bei zwei Laufwerken ein und Sie erhalten:

```
Dies  ist  Zeile  1.
Dies  ist  Zeile  2.
Dies  ist  Zeile  3.
Dies  ist  Zeile  4.
Dies ist Zeile 5.
```

Tatsächlich handelt es sich dabei um den eingegebenen Text, jedoch ohne Zeilennummern, die nicht mit abgespeichert werden. Die einzelnen Zeilen befinden sich nämlich kontinuierlich oder sequentiell auf der Diskette, wobei an jedem Zeilenende ein Carriage-Return- und ein Line-Feed-Zeichen ausgegeben wird.

Spaßeshalber wollen wir uns jetzt einmal das Directory ansehen und geben dazu

```
DIR PROBE.*
```

bzw.

```
DIR B:PROBE.*
```

ein. Und siehe da, die Textdatei ist nicht nur einmal, sondern sogar doppelt gespeichert, denn wir erhalten:

```
A: PROBE    BAK : PROBE    TXT
```

ED legt nämlich automatisch eine Sicherheits- oder Backup-Kopie mit gleichem Dateinamen unter der Extension BAK an. Die BAK-Datei ist zwar in unserem Fall noch leer, da wir den Text gerade neu angelegt haben. Wenn wir die Datei aber später ändern oder fortschreiben, wird die alte Orginaldatei zur BAK-Datei umbenannt, während die neue Datei die richtige Extension erhält. Auf diese Weise können wir immer noch auf eine Sicherheitskopie mit dem letzten Bearbeitungsstand zurück-greifen, falls der neu bearbeiteten Datei etwas zustoßen sollte.



### 4.2.2 Zeilen einfügen/löschen

Jetzt wollen wir daran gehen, unsere Textdatei PROBE.TXT zu verändern, indem wir Zeilen einfügen und löschen. Wenn wir dies beherrschen, besitzen wir schon einen guten Grundstock, beliebige Texte mit ED zu bearbeiten.

Um diesen Vorgang zu verstehen, müssen wir wissen, daß ED intern einen Textpuffer anlegt, der den zu bearbeitenden Text aufnimmt. Die Größe dieses Puffers ist jedoch begrenzt, so daß bei umfangreichen Dateien immer nur ein Teil in ihm abgelegt werden kann. Da es für solche Dateien bessere Editiermöglichkeiten als mit ED gibt und unser Beispieltext nur wenige Zeilen umfaßt, gehen wir im Augenblick davon aus, daß der gesamte Text aus der Orginaldatei in den Puffer geladen wird.

Während Sie mit ED arbeiten, wird eine neue Datei auf der Diskette an-gelegt, die man als Zwischendatei bezeichnet und die mit der Extension $$$ gekenzeichnet ist. Die Orginaldatei bleibt dabei völlig unberührt und wird nach Abschluß der Textverarbeitung in eine BAK-Datei umbenannt, während die Zwischendatei den eigentlichen Dateinamen erhält.

Auf unsere Datei PROBE.TXT bezogen bedeutet dies, daß während der Bearbeitung eine Zwischendatei PROBE.$$$ angelegt wird, in die der überarbeitete Text geschrieben wird. Wenn Sie dann die Bearbeitung abschließen, wird die ursprüngliche Datei in PROBE.BAK und die Zwischendatei in PROBE.TXT umbenannt.

Nachdem wir unsere Textdatei bereits auf Diskette mit dem E-Befehl gesichert haben, müssen wir ED erneut starten, um sie zu verändern. Geben Sie deshalb wieder

```
ED PROBE.TXT
```

bzw.

```
ED B:PROBE.TXT
```

ein, um ED zu laden und zu aktivieren. Daraufhin sucht der Editor das Directory nach dem angegebenen Dateinamen ab. In diesem Fall ist er bereits vorhanden, weshalb nun nicht mehr der Hinweis `NEW FILE`, sondern nur noch

erscheint und wir jetzt mit der eigentlichen Bearbeitung beginnen können. Doch zuvor dürfen wir nicht vergessen, die Textdatei in den Puffer zu laden, wozu wir

#A

eingeben. Das Laufwerk beginnt sich noch einmal zu drehen und lädt den Text ein. Wenn wir jetzt

B#T

eingeben, werden unsere fünf Textzeilen wieder aufgelistet:

```
: * #A                    (Text in Puffer lesen)
1: *B#T                   (Text von Beginn an ausgeben)
1: Dies ist Zeile 1.
2: Dies ist Zeile 2.
3: Dies ist Zeile 3.
4: Dies ist Zeile 4.
5: Dies ist Zeile 5.
1:*
```

Statt des Ziffernkreuzes (#) könnten wir auch einen Zahlenwert setzen. So würde

3A

nur drei Zeilen in den Textpuffer laden und

B2T

nur die ersten beiden Zeilen im Puffer listen. Übrigens wird mit

B

der interne Pufferzeiger auf den Pufferanfang und mit

—B

wieder auf das Pufferende gesetzt. Somit bedeutet

B#T

"Setze den Zeiger an den Pufferanfang und gib sämtliche Zeilen aus". Dabei wird die Textausgabe mit T gesteuert. B und #T hätten wir allerdings auch getrennt erteilen können, aber ED erlaubt uns, mehrere Befehle in einer Zeile hintereinander und ohne Zwischenraum anzugeben.



Den Pufferzeiger können wir auf jede beliebige Zeile setzen. Wenn wir beispielsweise

3:

eingeben, steht er am Anfang der dritten Zeile. Geben wir daraufhin

2T

ein, werden von dort aus zwei Zeilen ausgegeben und es erscheint

```
3: Dies ist Zeile 3.
4: Dies ist Zeile 4.
```

Beide Befehle hätten wir auch wieder gemeinsam angeben können und damit dasselbe Resultat erreicht:

3 : 2T

Erhält T einen negativen Zahlenwert, also beispielsweise

-2T

erscheinen die beiden Zeilen über dem Zeiger:

```
1: Dies ist Zeile 1.
2: Dies ist Zeile 2.
```

Fassen wir nochmals zusammen:

| | |
|---|---|
| nA | liest n Zeilen von Diskette in den Textpuffer |
| +/—B | setzt den Textpufferzeiger an den Textanfang bzw. das Textende |
| n: | setzt den Pufferzeiger an den Anfang der n-ten Zeile |
| +/—nT | listet n Zeilen ab bzw. vor dem Pufferzeiger |
| # | steht anstelle von n und bedeutet "sämtliche Zeilen" |
| +/¯nL | bewegt den Pufferzeiger von seiner gegenwärtigen Position n Zeilen auf- bzw. abwärts |

Jetzt wollen wir zwischen Zeile 2 und 3 zwei weitere Zeilen einfügen. Dazu müssen wir den Pufferzeiger auf die dritte Zeile setzen, denn die neuen Zeilen erscheinen ja vor dieser Zeile, weshalb sämtliche Zeilen ab der dritten Zeile beim Einfügen nach unten verschoben werden:

```
1: *3:                    (Pufferzeiger auf dritte Zeile)
3 : *i                    (Einfügemodus einschalten)
3: Dies ist die neue Zeile 2a.
4: Dies ist die neue Zeile 2b.
```

```
5:                                  (hier <CTRL¯Z> eingeben)
5:  *
```

Nachdem nun der Pufferzeiger am Anfang der dritten Zeile steht, muß mit i wieder der Einfügemodus eingeschaltet werden. Wir geben die beiden neuen Zeilen wie gewohnt ein und drücken abschließend <CTRL-Z>.

Nun kontrollieren wir, ob die neuen Zeilen auch richtig eingefügt sind:

```
1: *B#T                 (Zeiger an Textanfang und alle Zeilen listen)
1: Dies ist Zeile 1.
2: Dies ist Zeile 2.
3: Dies ist die neue Zeile 2a.
4: Dies ist die neue Zeile 2b.
5: Dies ist Zeile 3.
6: Dies ist Zeile 4.
7: Dies ist Zeile 5. 1: E       (Datei auf
Diskette sichern)
```

Da die neuen Zeilen an der richtigen Stelle stehen, haben wir auch gleich den Text mit E auf Diskette gesichert und den Editor verlassen. Wir können jetzt, wenn es uns interessiert, die geänderte Datei mit TYPE ansehen, wie wir es nach ihrer Erstellung bereits getan haben.

Nachdem uns nun das Einfügen der beiden Zeilen gelungen ist, wollen wir sie zu Übungszwecken gleich wieder löschen. Dazu laden wir den Editor erneut unter Angabe des Dateinamens PROBE.TXT. Diesen Vor-gang könnten wir uns aber ersparen, wenn wir den Editor vorher nicht mit E verlassen hätten. Dann wäre auch der Text noch im Puffer, den wir in unserem Fall mit **#A** erst wieder neu einladen müssen.

Wir setzen den Pufferzeiger auf die dritte Zeile, die nun die erste Zeile darstellt, die es zu löschen gilt. Mit

```
2K
```

werden die beiden Zeilen gelöscht, wobei K (engl. Kill) für Löschen und die Ziffer 2 für die beiden Zeilen steht. Abschließend listen wir den Text nochmals auf. Hier nun alles, was sich auf dem Bildschirm abspielt:

```
  * #A                  (Text in Puffer lesen)
1: *3:                  (Pufferzeiger auf 3. Zeile)
3: *2K                  (von da ab 2 Zeilen löschen)
3: B#T                  (Text auflisten)
1: Dies  ist Zeile 1.
```



```
2: Dies  ist Zeile  2.
3: Dies  ist Zeile  3.
4: Dies  ist Zeile  4.
5: Dies  ist Zeile  5.
1: *
```

### 4.2.3 **Text korrigieren**

Im letzten Abschnitt haben wir gelernt, Textzeilen einzufügen und zu löschen. Für viele Korrekturen wird es die einzige Möglichkeit bleiben, die gesamte fehlerhafte Zeile zu löschen und im Einfügemodus wieder neu zu schreiben und in den Text einzufügen. Wesentlich komfortabler wäre es allerdings, eine oder mehrere Zeilen aufzulisten, mit dem Cursor an die fehlerhafte Stelle vorzurücken und diese dann auszubessern.

Leider müssen wir bei ED auf diese Möglichkeit verzichten, weshalb sie nur Textverarbeitungsprogrammen wie WORDSTAR vorbehalten bleibt. Den einzigen Luxus, den ED bietet, ist das Suchen und Austauschen von Wörtern im Text.

Haben wir nun in einer Textzeile einen Fehler festgestellt, müssen wir ED das fehlerhafte Wort erst suchen lassen, worauf es gelöscht und durch ein neues ersetzt wird. Dies ist wirklich eine sehr umständliche Methode, Texte zu korrigieren, aber einfacher geht es leider nicht, es sei denn, man ersetzt die gesamte Zeile.

Diese Art der "Korrektur" wollen wir nachfolgend anhand eines Beispiels ausprobieren. Laden Sie dazu erneut den Editor unter Angabe der Datei PROBE.TXT und lesen Sie diese wieder mit **#A** in den Textpuffer. Wenn Sie jetzt

```
2:T                 (Zeiger auf Zeile 2 und diese listen)
```

eingeben, erhalten Sie den Text, den wir ursprünglich eingegeben haben:

```
2:      Dies ist Zeile 2.
```

Jetzt wollen wir das Wort "Dies" durch "Hier" ersetzen, was folgendermaßen geschieht:

```
*sDies^ZHier^ZOLT
```

Daraufhin erscheint die korrigierte Zeile

```
2:      Hier ist Zeile 2.
```

Lassen Sie uns die Befehlszeile einmal etwas genauer betrachten. Hinter dem Sternchen, das ja ohnehin zur Befehlseingabe auffordert, steht ein kleines s. Unter s versteht ED "Ersetze" (engl. Substitute). Bei den beiden nachfolgenden Wörtern handelt es sich zuerst um das alte und schließlich um das neue Wort, wobei beide Wörter mit **^Z (<CTRL—Z>)** abschließen **müssen.** Achten Sie auch hier wieder darauf, daß wir ein kleines s angeben. Bei einem Großbuchstaben würde, ähnlich wie bei I (Einfügen), nicht "Hier", sondern "HIER" im korrigierten Text er-scheinen, selbst wenn wir Kleinbuchstaben eingeben.

Mit OLT (erstes Zeichen eine Null, kein Buchstabe 0!) erhält ED schließlich die Anweisung, den Pufferzeiger zurück an den Zeilenanfang zu setzen und die Zeile dann auszugeben. Somit haben wir gleichzeitig die Möglichkeit, festzustellen, ob die Korrektur richtig stattgefunden hat.

ED besitzt noch einige weitere Befehle als diejenigen, die wir bisher kennengelernt haben. Sie sind in der Zusammenfassung aller CP/M-Befehle in Kapitel 10 aufgeführt. Auf ihre ausführliche Behandlung wollen wir verzichten, da sie an dieser Stelle mehr Verwirrung als Klarheit schaffen würden. Mit den hier beschriebenen Befehlen sollten Sie aber in der Lage sein, einfache Texte einzugeben und zu korrigieren, sofern Sie nicht ohnehin schon auf andere Textverarbeitungsprogramme zurückgreifen.

Übrigens kann die Datei PROBE.TXT, die wir mit ED auf Diskette erzeugt haben, auch von BASIC aus ganz gewöhnlich als sequentielle Datei gelesen werden. Wir können sogar so weit gehen, daß wir ein ganzes BASIC-Programm mit ED oder einem anderen Textverarbeitungsprogramm schreiben, es als BAS-Datei auf Diskette ablegen, um es dann von BASIC aus auszuführen. Eine so erzeugte Datei entspricht dann genau einem Programm, das in BASIC mit SAVE "Name" , A abgespeichert wurde. Umgekehrt kann natürlich auch ein BASIC-Programmlisting in eine Textdatei übernommen werden, wenn es sich im ASCII-Code auf der Diskette befindet.



# 5 PIP - ein universelles Kopierprogramm

## 5.1 Allgemeines

PIP, englisch "Peripherial Interchange Processor" ist ein weiteres Standardprogramm für jedes CP/M-System. Um ihm eine allgemeine Kurzbezeichnung zu geben, haben wir es in der Überschrift als universelles Kopierprogramm bezeichnet. Dieser Begriff ist zwar durchaus zutreffend, beschreibt aber nicht sämtliche Möglichkeiten, die uns PIP bietet.

Für den CPC haben wir bereits einige andere Kopierprogramme, wie z.B. DISCCOPY, COPYDISC und FILECOPY kennengelernt, die größtenteils ähnliche Eigenschaften wie PIP besitzen, denn alles, was diese Programme kopieren, kann man mit PIP ebenfalls kopieren. PIP kann sogar die Aufgaben von CHKDISC und DISCCHK übernehmen, um zu prüfen, ob eine Datei richtig übertragen wurde.

Zwei Dinge kann PIP jedoch nicht, nämlich Disketten formatieren und bei Verwendung von nur einem Laufwerk eine Datei auf eine andere Diskette kopieren. Dies wäre nur dann denkbar, wenn der CPC ein zweites virtuelles Laufwerk B unterstützen würde. Virtuell bedeutet hier, daß ein Laufwerk sowohl als Laufwerk A als auch als Laufwerk B eingesetzt werden kann, wobei natürlich ständig die Disketten gewechselt werden müßten, ähnlich wie es bei den oben genannten Kopierprogrammen der Fall ist. Eine solche Möglichkeit müßte allerdings im BIOS verankert sein, so daß vor jedem Zugriff auf das andere Laufwerk eine Meldung ausgegeben wird, die den Benutzer auffordert, die Diskette zu wechseln.

In der Tat gibt es einige CP/M 2.2-Computer, die ein virtuelles Laufwerk B unterstützen, aber leider gehört der CPC nicht dazu. Lediglich CP/M Plus sieht diese Möglichkeit vor, das aber nur auf dem CPC 6128 und nicht auf dem CPC 464 und 664 lauffähig ist. Das ständige Wechseln der Disketten ist auf die Dauer gesehen jedoch sehr lästig und sollte wirklich

nur als Notbehelf angesehen werden. Deshalb sollten Sie sich, wenn irgend möglich, ein echtes zweites Laufwerk zulegen, das nicht nur unter PIP ein erheblich komfortableres Arbeiten ermöglicht.

Von den beiden genannten Einschränkungen einmal abgesehen, können Sie mit PIP wesentlich vielseitiger kopieren als z.B. mit FILECOPY. So ist es beispielsweise damit möglich, Dateien aneinanderzuhängen oder Auszüge von ihnen herzustellen. Dabei kann die Zieldatei durchaus auf das gleiche Laufwerk kopiert, muß dort aber unter einem anderen Namen abgelegt werden. Außerdem können Sie Dateien nicht nur von Diskette zu Diskette kopieren, sondern auch über jeden anderen Kanal, wie z.B. den Drucker oder die serielle Schnittstelle, ausgegeben. Die vier Ein- und Ausgabekanäle haben wir im Zusammenhang mit STAT (siehe Kapitel 3) bereits kennengelernt.

PIP wurde zwar in erster Linie zum Kopieren und zur Übertragung von Textdateien im ASCII-Code entwickelt, es kann aber durchaus auch mit anderen Dateien, die einen beliebigen Binärcode enthalten, arbeiten, was die universellen Einsatzmöglichkeiten steigert. Wir werden in diesem Kapitel auch eine Vielzahl von Optionen kennenlernen,.. die beim Kopieren berücksichtigt werden und eine Datei während der bertragung zusätzlich verändern. So kann z.B. eine Textdatei in Groß- oder Kleinbuchstaben umgeformt werden oder bestimmte Steuerzeichen für den Ausdruck erhalten.

## 5.2 Laden und Aktivieren von PIP

PIP ist eine auf der Systemdiskette abgelegte COM-Datei. Wenn wir nun mit diesem Programm arbeiten wollen, gibt es zwei Möglichkeiten. Entweder wir starten es und geben gleichzeitig Anweisungen, was wir kopieren bzw. übertragen wollen. Allgemein ausgedrückt schreiben wir dafür

```
PIP (Befehlsfolge)     <ENTER>
```

PIP führt nun sämtliche Befehle aus und führt anschließend einen Warmstart durch, ähnlich wie es auch bei STAT der Fall ist. Diese Form des Aufrufs hat jedoch den Nachteil, daß PIP während des Kopiervorganges immer auf der Diskette in Laufwerk A oder B abgelegt sein muß, was in manchen Fällen sicher sehr lästig ist.



Wir können PIP aber auch ohne Befehlsfolge eingeben, was folgendermaßen geschieht:

```
PIP <ENTER> *
```

PIP ist jetzt geladen und aktiviert und befindet sich nun im Befehlsmodus, was wir an dem ausgegebenen Sternchen erkennen. Wir können jetzt die Diskette, die PIP enthält, aus dem Laufwerk herausnehmen und statt dessen diejenige einlegen, auf die bzw. von der kopiert werden soll. Dabei geben wir dieselbe Befehlsfolge wie im ersten Fall ein.

Nach Ausführung der Befehle kehrt PIP wieder in den Befehlsmodus zurück und ein weiteres Sternchen erscheint auf dem Bildschirm. Wir können nun auf die gleiche Weise weitere Befehle erteilen, die dann jeweils ausgeführt werden. Diesen Vorgang können wir beliebig oft wiederholen, bis wir unsere Arbeit mit PIP beendet haben. Dann drücken wir die ENTER-Taste, worauf PIP verlassen und ein Warmstart durchgeführt wird.

Gerade für die Anwender, die mit einem Laufwerk auskommen müssen, dürfte die letzte Methode, bei der PIP nicht erst kopiert werden muß, wesentlich angenehmer sein.

## 5.3 Kopieren von Diskette zu Diskette

In diesem Abschnitt wollen wir den einfachsten und naheliegendsten Einsatz von PIP betrachten, nämlich das Kopieren von Dateien auf die gleiche oder eine andere Diskette. Zum Übertrag auf eine andere Diskette ist jedoch ein zweites Laufwerk B erforderlich.
Die allgemeine Kopieranweisung lautet hier:

```
PIP [Laufwerk:]Zieldatei=[Laufwerk:]Quelldatei
```

bzw.

```
* [Laufwerk:]Zieldatei=[Laufwerk:]Quelldatei
```

Im ersten Fall wird PIP aufgerufen, wobei gleichzeitig der Kopiervorgang in die Wege geleitet wird, während im zweiten Fall PIP bereits geladen und aktiviert ist, so daß wir nur noch die Befehlszeile eingeben müssen.

Das Laufwerk muß nur angegeben werden, wenn mit einem anderen als dem Bezugslaufwerk gearbeitet wird. Haben Sie ohnehin nur ein Lauf-werk zur Verfügung, können Sie auf diese Angabe verzichten.

Eine Quelldatei ist immer diejenige Datei, die kopiert werden soll, während die Zieldatei der kopierten Datei entspricht.

Als Anwendungsbeispiel nehmen wir einmal an, Sie möchten von der Datei MUSTER.TXT eine Backup-Kopie mit der Extension BAK auf der gleichen Diskette herstellen, da dies anderweitig noch nicht geschehen ist. Wir gehen davon aus, daß nur ein Laufwerk angeschlossen ist.

Angenommen, PIP befindet sich auf der gleichen Diskette wie MUSTER. TXT, dann können wir die Befehlsfolge gleich mit angeben:

PIP **MUSTER.BAK=MUSTER.TXT** <ENTER>

PIP wird geladen, worauf sich unmittelbar der Kopiervorgang anschließt.

Enthält dagegen die Diskette nicht das PIP-Programm, legen wir zunächst die Systemdiskette ein und rufen PIP ohne Befehlsfolge auf:

PIP <ENTER> *

Nachdem das Sternchen erschienen ist, legen wir die Diskette mit MUSTER.TXT ein und schreiben erst dann die Befehlszeile:

*MUSTER.BAK=MUSTER.TXT <ENTER>

Dabei dürfen Sie natürlich nicht das Sternchen mit eintippen, denn es wird ja ohnehin ausgegeben.

Ist der Kopiervorgang beendet, erscheint in der nächsten Zeile ein weiteres Sternchen:

*

Wenn Sie jetzt weiter mit PIP arbeiten wollen, geben Sie den nächsten Befehl ein, wenn nicht, drücken Sie die ENTER-Taste, worauf Sie PIP wieder verlassen.

Bisher haben wir diese ausführliche Darstellungsform gewählt, damit Sie lernen, mit PIP umzugehen. Künftig geben wir nur noch die Befehle als solche an, also ohne PIP und ohne Sternchen, denn Sie wissen ja jetzt, wie PIP zu starten ist.



Das nächste Beispiel ist für Anwender mit zwei Laufwerken bestimmt. Wir wollen die Datei MUSTER.TXT von Laufwerk A nach Laufwerk B übertragen, wo wir sie unter dem gleichen Namen ablegen. Hierzu lautet der Befehl:

B: **MUSTER.TXT=A:MUSTER.TXT**

Wie Sie sehen, muß dieses Mal das Laufwerk mit angegeben werden. Selbstverständlich hätten wir die Datei auch unter einem anderen Namen in Laufwerk B ablegen können. Hier haben wir aber den gleichen Namen beibehalten und könnten deshalb die Befehlszeile auch kürzer schreiben:

**B:=A:MUSTER.TXT**

Ist nämlich für das Ziellaufwerk kein Name angegeben, wird automatisch der ursprüngliche Dateiname übernommen.

Auch PIP läßt mehrdeutige Dateinamen zu, so daß wir ganze Dateigruppen mit einem Befehl kopieren können. Es gelten dabei die gleichen Regeln, die wir schon in Kapitel 1 im Zusammenhang mit dem DIR-Befehl kennengelernt haben. Auch dazu zwei Beispiele:

A:=B:*.COM

kopiert sämtliche COM-Dateien von Laufwerk B nach Laufwerk A und

B:=A:*.*

kopiert sämtliche Dateien, die sich auf Laufwerk A befinden, nach Laufwerk B.

Indem Sie PIP auf diese Weise einsetzen, können Sie, sofern Sie zwei Laufwerke angeschlossen haben, durchaus auf das Programm FILECOPY verzichten. FILECOPY kann nämlich nur mit einem Laufwerk arbeiten, wobei unter Umständen die Diskette mehrfach gewechselt werden muß, was bei PIP entfällt.

## 5.4 Dateien aneinanderhängen

PIP bietet die Möglichkeit, mehrere Dateien zu einer neuen Datei zu vereinigen. Wir müssen uns dabei allerdings überlegen, in welchen Fällen eine solche Maßnahme auch tatsächlich sinnvoll ist.

Wenn Sie z.B. Maschinenprogramme, wie es die COM-Dateien in der Regel sind, miteinander verknüpfen, ist das sicher nicht sinnvoll! Jedes Maschinenprogramm (Objektcode) enthält nämlich im allgemeinen viele absolute Adressen, die sich auf den Speicherbereich beziehen, in dem das Programm abgelegt und abgearbeitet wird. Durch die Verknüpfung verschieben sich diese Adressen mit Sicherheit, so daß das Programm gar nicht mehr ausführbar ist und der Rechner wahrscheinlich abstürzt.

Vielleicht wäre ein Aneinanderreihen von BASIC-Programmen, die im internen BASIC-Code gespeichert sind, bedingt sinnvoll, wenn die einzelnen BASIC-Zeilen anschließend neu gelinkt (verbunden) werden. Aber hierzu gibt es andere und bessere Möglichkeiten.

Wirklich empfehlenswert und nützlich ist diese Methode nur bei Textdateien im ASCII-Code. Aus Kapitel 4, in dem wir uns mit dem Editor beschäftigt haben, wissen wir, daß alle Informationen, die der Computer aufnimmt, letztlich nur in Form von ASCII-Zeichen darstellbar sind. So können Sie ein BASIC-Programm als ASCII-Datei speichern, die dann wieder als ganz gewöhnlicher Text gelesen werden kann. Auch Maschinenprogramme werden meist zuerst im Assembler-Quelicode geschrieben, der noch unabhängig vom Speicherbereich ist und ebenfalls einen Text aus ASCII-Zeichen darstellt. Ganz zu schweigen von "normalen" Texten, zu denen auch die Zeilen gehören, die Sie gerade lesen. Sie sind zwar der Rechtschreibung der deutschen Sprache, aber ansonsten keiner besonderen Syntax zum Zweck einer weiteren Umwandlung unterlegen. Alle derartigen Texte können Sie beliebig kombinieren und aneinanderhängen.

Reine Textdateien, die nur aus ASCII-Zeichen bestehen, sollten immer mit dem ASCII-Code IAH bzw. `<CTRL—Z>` abschließen, was normalerweise automatisch von den Texteditoren durchgeführt wird. Beim Lesen des Textes wird dann immer dieses Zeichen abgefragt, um das Textende festzustellen, denn der letzte Record oder Block dieser Datei ist meistens nur zum Teil beschrieben und enthält noch freien Speicherplatz.

Wenn PIP nur eine Datei kopiert, fragt es den Code 1AH nicht ab, denn es überträgt immer den ganzen letzten Block, selbst wenn dieser nicht vollständig belegt ist. Deshalb können wir beispielsweise auch COMDateien korrekt übertragen. Anders ist es, wenn Dateien verknüpft wer-den sollen, da in diesem Fall das genaue Ende festgestellt werden muß. Aus diesem Grund verbindet PIP nur Dateien, von denen es annimmt, daß sie ASCII-Code enthalten. Befindet sich dagegen in einem Maschinen- oder BASIC-Programm an irgendeiner Stelle der Code IAH, was durchaus wahrscheinlich ist, wird die Übertragung an dieser Stelle abgebrochen, da PIP fälschlicherweise annimmt, eine Textdatei sei zu Ende.



Wenn wir nun Textdateien miteinander verbinden, müssen wir nur eine Zieldatei, aber mehrere durch Komma getrennte Quelldateien in der Befehlszeile angeben. Vielleicht haben Sie noch die Textdatei PROBE.TXT, die wir mit dem Editor in Kapitel 4 erstellt haben, auf Ihrer Diskette. Diese Datei wollen wir nun dreimal aneinanderhängen und daraus eine neue Datei mit dem Namen PROBENEU.TXT erzeugen. Wir geben dazu folgenden Befehl ein:

```
PROBENEU.TXT=PROBE.TXT, PROBE .TXT,PROBE .TXT
```

Befinden sich die Zieldatei und die zu verknüpfenden Dateien in verschiedenen Laufwerken oder aber allesamt in Laufwerk B, müssen wir vor jedem Dateinamen noch die Laufwerksangabe setzen, wie z.B.:

```
A:PROBENEU.TXT=B:PROBE.TXT,B: PROBE.TXT,B:PROBE.TXT
```

oder wenn verschiedene Quelldateien verwendet werden, `A:NEUDATEI.TXT`

```
=A:DATEI.BAK,B:BRIEF.TXT,B:LAGER.DAT
```

Hier befindet sich die Zieldatei NEUDATEI.TXT und die Quelldatei DATEI. BAK in Laufwerk A, während die anderen beiden Quelldateien BRIEF.TXT und LAGER.DAT in Laufwerk B abgelegt sind.

## 5.5 Datenaustausch mit anderen Peripheriegeräten

PIP kann Dateien nicht nur von Diskette zu Diskette übertragen, sondern auch von Diskette auf Peripheriegeräte und umgekehrt.

Lassen Sie uns den Begriff Peripherie einmal etwas genauer betrachten. Darunter fällt jedes externe Gerät, das an die Zentraleinheit angeschlossen ist, also neben der Floppy, ohne die ja CP/M gar nicht betrieben werden kann, auch Drucker, Bildschirm und Tastatur (Terminal) sowie andere Zusatzgeräte wie beispielsweise Modem (einschließlich serieller Schnittstelle) oder Harddisc.

Auf all diese Geräte kann PIP Dateien ausgeben. In einigen Fällen, z.B. über ein Modem, können Dateien auch eingelesen und auf Diskette gespeichert werden. Dabei handelt es sich zumeist um reine ASCII-Dateien, besonders wenn Text auf dem Drucker oder auf dem Bildschirm ausgegeben wird.

Obwohl Ihr CPC ohnehin standardmäßig bereits mit Bildschirm, Tastatur und eventuell Drucker ausgestattet ist, betrachtet PIP diese Geräte als externe Peripherie.

Als wir uns in Kapitel 3 mit dem STAT-Befehl befaßten, sprachen wir auch darüber, daß CP/M vier Ein- und Ausgabekanäle zum Datenaustausch mit externen Geräten benutzt. Diese Kanäle sind standardmäßig vorbelegt, wobei das IOBYTE die jeweilige Gerätekonfiguration enthält und gegebenenfalls geändert werden kann.

Zur Wiederholung führen wir nochmals den STAT DEV:-Befehl aus und erhalten:

```
CON: is CRT:    (Terminalkanal)
RDR: is TTY:    (
PUN: is TTY:    (Lochstreifenleserkanal
LST: is LPT:    (Lochstreifenstanzerkan
                (Druckerkanal)
```

Links stehen die vier Ein- und Ausgabekanäle und rechts die im IOBYTE zugeordneten Geräteeinheiten.

Nun können wir auch PIP-Befehle angeben, die anstelle einer Ein-/Ausgabedatei einen dieser Kanäle angibt. Den wohl häufigsten Anwendungsfall stellt die Ausgabe von Textdateien auf den Drucker oder auf den Bildschirm dar.

Sie werden sich jetzt vielleicht sagen, daß dies auch mit dem TYPE-Befehl erreicht werden kann, was durchaus richtig ist. Mit PIP stehen uns aber noch weitere Variationsmöglichkeiten zur Verfügung, die wir in diesem und im nächsten Abschnitt kennenlernen werden.

Grundsätzlich gilt auch bei der Einbeziehung von externen Geräten die Befehlsfolge

```
Zielgerät=[Laufwerk:]Quelldatei
```

bzw.

```
[Laufwerk:]Zieldatei=Quellgerät
```

d.h. es ist immer zuerst die Zieleinheit und dann, durch ein Gleichheitszeichen getrennt, die Quelleinheit anzugeben.

Jetzt wollen wir einmal unsere Beispieldatei PROBE.TXT auf dem Bildschirm auflisten, wozu wir folgendes eingeben:

```
CON:=PROBE.TXT
```



bzw.

```
CON:=B:PROBE.TXT
```

Dabei gilt der letzte Befehl, wenn die Datei von Laufwerk B, das nicht Bezugslaufwerk ist, gelesen werden soll. Hierin ist CON: (=Bildschirm) das Zielgerät, das anstelle einer Zieldatei aufgeführt ist.

Dasselbe wollen wir jetzt auch einmal mit dem Drucker ausprobieren, was wir mit folgender Anweisungen erreichen:

```
LST:=PROBE.TXT
```

bzw.

```
LST:=B:PROBE.TXT
```

Wir können natürlich auch die Datei PROBE.TXT dreimal hintereinander auf dem Drucker ausgeben:

```
LST:=PROBE .TXT, PROBE .TXT,PROBE .TXT
```

Falls wir über den TTY-Kanal eine serielle Schnittstelle angeschlossen haben, können wir mit

```
PUN: =MODEM. TXT
```

die Datei MODEM.TXT über die Schnittstelle ausgeben bzw. sie mit

```
MODEM. TXT=RDR:
```

über dieselbe einlesen.

Zum Abschluß betrachten wir noch ein interessantes Beispiel, in dem wir direkt über die Tastatur in eine Datei schreiben. Versuchen Sie einmal folgendes:

```
TERMINAL.TXT=CON:
```

Wenn jetzt der Cursor am Anfang der nächsten Zeile erscheint, tippen Sie einen kurzen Text ein, wie z.B.:

```
Dies ist die Datei TERMINAL.TXT. <CTRL-Z>
```

Achten Sie unbedingt darauf, daß Sie am Schluß <CTRL-Z> drücken, womit das Dateiende bei Textdateien gekennzeichnet wird. Die Datei wird dann geschlossen und PIP wartet auf den nächsten Befehl, bzw. das Bereitschaftszeichen A> erscheint wieder auf dem Bildschirm.

Denken Sie aber nicht, daß diese Methode zur Textverarbeitung geeignet ist und das Programm ED oder einen sonstigen Texteditor überflüssig macht! Sie haben hier nämlich keinerlei Möglichkeit, den eingegebenen Text zu korrigieren, da er direkt in die Datei geschrieben wird. Außer-dem müssen Sie an jedem Zeilenende die Zeichen Carriage-Return und Line-Feed von Hand eingeben, was mit der ENTER-Taste bzw. <CTRLM> und <CTRL—J> geschieht.

## 5.6 Spezielle Befehle

Wir haben nun die wichtigsten PIP-Befehle kennengelernt, die für den Alltagsgebrauch eigentlich ausreichen müßten. Es gibt aber noch eine Reihe von Spezialbefehlen und Optionen, von denen einige nachfolgend vorgestellt werden sollen. In der Zusammenfassung in Kapitel 10 sind je-doch sämtliche Befehle noch einmal aufgeführt.

### 5.6.1 Dateiausschnitte

Mit PIP könnnen Sie auch Teile einer Textdatei übertragen. Dies funktioniert ähnlich, als wenn Sie einen mit ED erstellten Text korrigieren möchten, denn Sie müssen dabei ein Suchkriterium vorgeben. Die Datei wird dann von bzw. ab der Stelle, an der der gesuchte Begriff auftritt, übertragen.

Als Beispiel verwenden wir wieder unsere Datei PROBE.TXT, die wir mit ED erstellt haben. In ihrer ungekürzten Form hat sie folgenden Inhalt:

```
Dies ist Zeile 1.
Dies ist Zeile 2.
Dies ist Zeile 3.
Dies ist Zeile 4.
Dies ist Zeile 5.
```

Jetzt wollen wir diese Datei ab der Stelle auf dem Bildschirm ausgeben, an der der Begriff "Zeile 3." auftritt. Da dieser Begriff Groß- und Klein-buchstaben enthält, müssen wir zunächst einmal PIP separat starten und dann die Befehlszeile eingeben. Würden wir ihn dagegen vom CCP aus in der Form von PIP (BEFEHLSZEILE) suchen, würden sämtliche Zeichen des Begriffs automatisch in Großbuchstaben umgewandelt, wodurch er natürlich nicht aufgefunden werden kann.



Starten Sie nun also PIP, falls Sie es noch nicht getan haben und geben Sie, wenn das Sternchen erscheint, folgendes ein:

CON:=PROBE.TXT[SZeile 3.^Z] Nun

erscheint

```
Zeile 3.
Dies ist Zeile 4.
Dies ist Zeile 5.
```

Im nächsten Beispiel geben wir die gleiche Datei aus, allerdings nur bis zu der Stelle, an der "Zeile 3." auftritt:

CON:=PROBE.TXT[QZeile 3.^Z]

Diesmal erhalten wir

```
Dies ist Zeile 1.
Dies ist Zeile 2.
Dies ist Zeile 3.
```

Selbstverständlich könnten wir hier statt des Bildschirms auch eine Diskettendatei als Ziel angeben, oder warten, bis ein bestimmter Begriff über ein Modem eingeht. Entscheidend jedoch ist nur der Inhalt der eckigen Klammer, den wir uns einmal genauer ansehen wollen.

S bedeutet, daß die Übertragung erst dann beginnen soll, wenn der gesuchte Begriff auftritt, bei **Q** dagegen findet die Übertragung ab dem Dateianfang statt und wird an dieser Stelle abgebrochen. Der gesuchte Begriff wird allerdings in beiden Fällen mit übertragen, wie wir gesehen haben.

Hinter S bzw. **Q** muß der gesuchte Begriff stehen, der mit CTRL—Z abgeschlossen sein muß. Fassen wir diese beiden Optionen noch einmal in allgemeiner Form zusammen:

| | |
|---|---|
| [S"BEGRIFF" ^Z ] | Übertragung bei "Begriff' beginnen |
| [Q" BEGRIFF" ^Z ] | Übertragung bei "Begriff" abbrechen |

### 5.6.2 **Groß- und Kleinschrift**

Die folgenden beiden Optionen ermöglichen eine Übertragung von Text-dateien, wobei sämtliche Buchstaben entweder in Groß- oder in Kleinschrift umgewandelt werden. Dabei bedeutet

[U] Umwandlung in Großschrift und
[L] Umwandlung in Kleinschrift

Wir wollen dies wieder mit unserer Datei PROBE.TXT ausprobieren und geben dazu

```
CON:=PROBE .TXT[U]
```

ein. Wie wir sehen, erscheinen nur große Buchstaben auf dem Bildschirm:

```
DIES IST ZEILE1.
DIES IST ZEILE2.
DIES IST ZEILE3.
DIES IST ZEILE4.
DIES IST ZEILE5.
```

Mit

```
CON:=PROBE.TXT[L]
```

dagegen werden alle Groß- in Kleinbuchstaben umgeformt:

```
dies ist zeile 1.
dies ist zeile 2.
dies ist zeile 3.
dies ist zeile 4.
dies ist zeile 5.
```

### 5.6.3 Textausgabe mit Zeilennummern

Wir haben die Möglichkeit, einen Text so zu übertragen, daß am Anfang jeder Zeile eine Zeilennummer steht. Strenggenommen wird die Zeilennummer immer dann eingefügt, wenn im Text ein Carriage-Return- und Linefeed-Zeichen auftritt. Die benötigten Optionen lauten hier

[N] für fortlaufende Zeilennummern und
[N2 ] für fortlaufende Zeilennummern (sechsstellig mit führenden Nullen)

Auf unsere Datei PROBE.TXT bezogen erscheint infolge von

```
CON:=PROBE .TXT[N]
```

folgende Bildschirmausgabe:

```
1: Dies ist Zeile 1.
2: Dies ist Zeile 2.
3: Dies ist Zeile 3.
4: Dies ist Zeile 4.
5: Dies ist Zeile 5.
```



Geben wir dagegen CON:=PROBE .TXT[N2] vor, erhalten wir führende Nullen:

```
000001 Dies ist Zeile 1.
000002 Dies ist Zeile 2.
000003 Dies ist Zeile 3.
000004 Dies ist Zeile 4.
000005 Dies ist Zeile 5.
```

Wir können auch mehrere Optionen gleichzeitig in eckiger Klammer angeben, wie z.B.:

```
CON:=PROBE .TXT[N2U]
```

Hier wird die Datei PROBE.TXT in Großbuchstaben und mit Zeilen-nummern einschließlich führender Nullen auf dem Bildschirm aus-gegeben.

### 5.6.4 Verify

Sie können, wenn Sie Dateien von Diskette zu Diskette übertragen, mit der V-Option gleichzeitig ein Verify durchführen. Dabei wird die Zieldatei normal auf Diskette geschrieben und anschließend überprüft, ob die Übertragung richtig stattgefunden hat. Trat dabei ein Fehler auf, er-scheint die Meldung:

```
VERIFY ERROR (Befehlszeile)
```

Hier ein Beispiel, in dem sämtliche Dateien von Laufwerk A nach Laufwerk B mit Verify übertragen werden:

```
B:=A:*.* [V]
```

Eine weitere Prüfung können Sie bei Textdateien mit der Echofunktion (Option E) durchführen. Dabei wird die Datei während des Kopierens auf dem Bildschirm ausgegeben. Auch hierzu ein Beispiel:

```
B:=A:PROBE.TXT[E]
```

### 5.6.5 Kopieren geschützter Dateien

Wollen Sie in eine Datei kopieren, die schreibgeschützt ist (R/0), erfolgt zunächst eine Sicherheitsabfrage:

```
DESTINATION IS R/0, DELETE (Y/N)?
```

Nur wenn Sie mit Y (Ja) antworten, können Sie in die Datei schreiben. PIP legt nämlich bei jedem Kopiervorgang auf Diskette zunächst eine Zwischendatei (Extension $$$) an. Erst anschließend wird eine eventuell vorhandene Datei unter gleichem Namen gelöscht und die Zwischendatei umbenannt.

Systemdateien werden normalerweise nicht von PIP kopiert, es sei denn, man gibt die R-Option an, wie z.B.:

```
B:=A:GEHEIM.DAT[R]
```

### 5.6.6 Zusammenfassen binärer Dateien

Aus Abschnitt 5.4 wissen wir, daß PIP nur solche Dateien zusammenfaßt, die mit CTRL—Z enden, was in der Regel bei Textdateien der Fall ist. In speziellen Anwendungsfällen kann es manchmal erforderlich sein, die Abfrage auf CTRL—Z aufzuheben, so daß die gesamte physikalische Dateilänge übertragen wird. Im folgenden Beispiel werden mit der 0-Option drei Binärdateien zusammengefaßt.

```
GESAMT. BIN=DATEI1. BIN, DATEI2 . BIN, DATEI . BIN [0]
```

Werden Dateien nur einzeln übertragen, also nicht verkettet, kann man auf die 0-Option verzichten, da hier keine Abfrage auf CTRL—Z statt-findet. Dies gilt ganz besonders für das Kopieren von COM-Dateien.



# 6 Stapelverarbeitung

## 6.1 Allgemeines

Wir haben nun schon einen Großteil der verfügbaren CP/M-Befehle kennengelernt und können damit eine ganze Menge anfangen. Je mehr wir jedoch mit CP/M arbeiten, desto häufiger kommt es vor, daß wir bestimmte Befehlsfolgen immer wieder ausführen müssen, wobei wir jedesmal die Befehle einzeln eingeben und durch Drücken der ENTER-Taste abschließen.

Vielleicht sind Sie schon müde, weil Sie immer und immer wieder die gleichen Befehlsfolgen eingegeben haben. Sicher war das eine gute Übung für Sie, denn Sie wollen ja beim Studium dieses Buches CP/M kennenlernen. Da Sie aber nun schon (fast) ein Profi sind, fragen Sie sich sicherlich zu Recht, ob man bestimmte Befehlsfolgen, die sich ständig wiederholen, auch einfacher handhaben kann.

Genau damit wollen wir uns in diesem Kapitel beschäftigen! Wir lernen im folgenden nämlich die Stapelverarbeitung (engl. Batch Processing) kennen, bei der wir sämtliche Befehle auf einen Stapel (Befehlsdatei) legen, der dann abgearbeitet wird.

Dieser Vorgang ist sehr einfach und leicht zu verstehen. Wir schreiben alle Befehle zeilenweise in eine normale Textdatei, wie wir es bei ED bereits kennengelernt haben. Dann rufen wir nur noch das CP/MDienstprogramm SUBMIT auf, das die Befehle der Reihe nach abarbeitet.

## 6.2 SUBMIT

Zur besseren Veranschaulichung betrachten wir uns wieder ein einfaches Beispiel.

Angenommen, Sie möchten das Directory der Diskette auflisten, sich dann die Dateiattribute der Textdatei PROBE.TXT ansehen und schließlich diese Datei auflisten. Die Datei PROBE.TXT hatten wir bereits in Kapitel 4 angelegt und wollen sie hier für unsere Beispiele verwenden.

Diese drei aufgeführten Befehle legen wir zunächst in einer Textdatei ab. Da sie zur Stapelverarbeitung dient, muß ihr Name mit der Extension SUB gekennzeichnet sein. Deshalb geben wir der Datei den Namen LIST.SUB.

In Kapitel 4, in dem wir uns mit ED befaßten, wurden bereits verschiedene Möglichkeiten erörtert, wie eine Texdatei angelegt werden kann. Haben Sie beispielweise WORDSTAR zur Verfügung, können Sie damit die Datei schreiben. Da es sich jedoch nur um wenige Textzeilen handelt, wählen wir hier den CP/M-Editor ED und geben folgendes ein:

```
ED LIST.SUB
NEW FILE
: *I
1 : DIR
2 : STAT PROBE.TXT
3 : TYPE PROBE.TXT
4 :     <CTRL-Z>
  *E
```

Wir haben jetzt die Datei LIST.SUB mit folgenden Befehlen angelegt:

```
DIR
STAT PROBE.TXT
TYPE PROBE.TXT
```

Sollte Ihnen die Handhabung des Editors zu kompliziert erscheinen, können Sie die gleiche Datei auch von BASIC aus erzeugen, beispielsweise mit folgendem Programm:

```
10 OPENOUT "LIST.SUB"
20 PRINT #9, "DIR"
30 PRINT #9, "STAT PROBE.
TXT" 40 PRINT #9, "TYPE
PROBE.TXT" 50 CLOSEOUT
```

Achten Sie darauf, daß sich sämtliche Dateien und Programme, die hier benötigt werden, auf einer Diskette befinden. Dabei handelt es sich neben der Befehlsdatei LIST.SUB auch um die Dateien STAT.COM und SUBMIT.COM, die Sie gegebenenfalls von der Systemdiskette auf Ihre Arbeitsdiskette kopieren müssen. Verwenden Sie dazu am besten FILECOPY oder PIP. Da auf die Arbeitsdiskette Schreibzugriffe



vorgenommen werden, darf sie auf keinen Fall schreibgeschützt sein, weder mechanisch (Riegel am Diskettengehäuse) noch softwaremäßig (mit STAT).
Wenn Sie jetzt

```
SUBMIT LIST
```

eingeben, werden sämtliche Befehle ausgeführt, die in der Datei LIST.SUB abgelegt sind. Auf dem Bildschirm sieht dies genauso aus, als hätten Sie jeden Befehl einzeln eingegeben:

```
A>SUBMIT LIST
A>DIR
A: PROBE   TXT   : PROBE   BAK : SUBMIT   COM :
LIST SUB
A: $$$ SUB
A>STAT
          Bytes    Ext Acc
PROBE.     lk      1 R/W  A:PROBE.TXT
  TXT          On  A: 143k
  Recs
Dies ist Zeile 1.
Dies ist Zeile 2.
Dies ist Zeile 3.
Dies ist Zeile 4.
Dies ist Zeile 5.
A>
```

Für die Stapelverarbeitung können Sie auch zwei Laufwerke verwenden, jedoch muß sich die SUBMIT.COM-Datei immer in Laufwerk A befinden, das nicht schreibgeschützt sein darf. Alle anderen Dateien können sich in Laufwerk B befinden.

Wenn beispielsweise die Befehlsdatei LIST.SUB in Laufwerk **B** abgelegt ist, muß der SUBMIT-Aufruf folgendermaßen aussehen:

```
SUBMIT B: LIST
```

Wenn SUBMIT eine Befehlsfolge abarbeitet, wird zunächst die SUBDatei, in unserem Fall LIST.SUB, in eine Zwischendatei mit der Bezeichnung $$$. SUB kopiert, die auf der gleichen Diskette wie SUBMIT an-gelegt wird, also immer im Laufwerk A. Anschließend wird die Zwischendatei gelesen und jeweils der Befehl aus ihr herausgenommen, der gerade abgearbeitet vAerden soll. Wenn sämtliche Befehle ausgeführt sind, ist die Zwischendatei leer und wird automatisch wieder gelöscht.

Da unsere Befehlsdatei auch den DIR-Befehl enthält, erscheint selbstverständlich die $$$.SUB-Datei auch im Directory, wie aus der Auflistung ersichtlich ist.

Zusätzlich bietet SUBMIT noch die Möglichkeit, Variablen bzw. Parameter an die Befehlsdatei zu übergeben. Dadurch wird ein noch komfortableres Arbeiten ermöglicht.

Unsere Befehlsdatei enthält zwei Zeilen, in denen der Dateiname PROBE. TXT vorkommt. Wenn wir nun die gleichen Befehle mit einer anderen Datei ausführen wollen, müßten wir die Befehlsdatei jedesmal ändern. Dies können wir dadurch umgehen, daß wir für den Dateinamen eine Variable einsetzen. Die Befehlsdatei LIST.SUB sähe in diesem Fall so aus:

```
 DIR
 STAT $1
TYPE $1
```

Anstelle des Dateinamens haben wir hier $1 angegeben, was dem ersten übergebenden Parameter entspricht. Weitere Parameter wären entsprechend mit $2, $3 usw. gekennzeichnet. Wir übergeben hier aber nur einen Parameter, für den wir den Dateinamen PROBE.TXT setzen. Dann lautet der SUBMIT-Befehlsaufruf so:

```
SUBMIT LIST PROBE.TXT
```

Soll die Befehlsdatei beispielsweise nicht mit PROBE.TXT, sondern mit LAGER.DAT abgearbeitet werden, müßten wir folgende Befehlszeile eingeben:

```
SUBMIT LIST LAGER.DAT
```

Übergeben wir statt einem mehrere Parameter an die Befehlsdatei, die darin mit $l, $2, $3 usw. gekennzeichnet sind, werden die Parameter in der SUBMIT-Befehlszeile der Reihe nach aneinandergefügt:

```
SUBMIT SUB—Datei Parameterl Parameter2
Parameter3 ...
```

Hier ein weiteres Beispiel: Angenommen, wir wollen die Dateien PROBE. TXT, LISTE.TXT und BRIEF.TXT in BAK-Dateien umbenennen, ohne den REN-Befehl dreimal hintereinander aufrufen zu müssen. Dazu legen wir folgende Befehlsdatei unter dem Namen NEUNAME.SUB an:



```
REN   $1.BAK=$1.TXT
REN   $2.BAK=$2.TXT
REN $3.BAK=$3.TXT
```

Als Variable dient nur der Dateiname ohne Extension, da diese bereits in der Befehlsdatei angegeben ist. Die Umbenennung der drei Dateien findet dann mit folgendem SUBMIT-Befehl statt:

```
SUBMIT NEUNAME PROBE LISTE BRIEF
```

Wir könnten auch für die jeweilige Extension eine Variable angeben, wobei TXT gleich $4 und BAK gleich $5 entspricht. Die Befehlsdatei sähe dann so aus:

```
REN   $1.$5=$1.$4
REN   $2.$5=$2.$4
REN $3.$5=$3.$4
```

Sie muß mit fogendem SUBMIT-Befehl aufgerufen werden:

```
SUBMIT NEUNAME PROBE LISTE BRIEF TXT BAK
```

Was geschieht nun aber, wenn beispielsweise SUBMIT eine Zwischendatei umbenennen soll, deren Dateiname ebenfalls das Dollarzeichen enthält? Die Antwort lautet sehr einfach: Vor jedes Dollarzeichen wird ein weiteres gesetzt. Auch hierzu ein Beispiel, in dem die Befehlsdatei NEUNAME.SUB folgende Zeile enthält:

```
REN $1.TXT=$1.$$$$$$
```

SUBMIT wird jetzt beispielsweise so aufgerufen:

```
SUBMIT NEUNAME LAGER
```

In diesem Fall wird die Zwischendatei LAGER.$$$ in LAGER.TXT umbenannt. Die sechs Dollarzeichen, die in der Befehlsdatei stehen, entsprechen in Wirklichkeit nur den drei Zeichen für die Extension $$$.

Grundsätzlich können Sie auf diese Weise jeden Parameter an die Befehlsdatei übergeben. Achten Sie aber darauf, daß die Parameter selbst keine Leerzeichen enthalten, die beim SUBMIT-Aufruf als Trennzeichen interpretiert werden. Umgekehrt kann eine SUBMIT-Zeile nur richtig abgearbeitet werden, wenn die einzelnen Parameter durch Leerzeichen getrennt sind.

## 6.3 XSUB

Mit der bisher behandelten Befehlsdatei konnten wir immer nur Parameter an die eigentliche Befehlszeile übergeben, niemals aber an andere Programme, die durch die Befehlsdatei aufgerufen werden.

Angenommen, wir rufen den Editor ED mit einem bestimmten Dateinamen auf, eine Aufgabe, die wir durchaus mit SUBMIT durchführen könnten. Das Arbeiten mit ED ist aber erst dann sinnvoll, wenn wir nach dem Aufruf weitere Anweisungen, die z.B. Zeilen löschen oder einfügen, an den Editor übergeben. Mit einem einfachen SUBMIT-Befehl ist dies jedenfalls nicht möglich.

Weitere Fälle dieser Art finden wir bei PIP oder beim Debugger DDT vor, die ebenfalls erst nach dem Aufruf Befehle entgegennehmen.

Um diesem Mißstand abzuhelfen, wurde den neueren CP/M-Versionen das Dienstprogramm XSUB beigefügt. Wenn XSUB die erste Zeile in einer Befehlsdatei einnimmt, können wir Variablen auch an andere Programme übergeben, damit sie damit arbeiten können. Diesen Vorgang wollen wir uns anhand eines praktischen Beispiels einmal näher an-schauen. Doch zuvor vergessen Sie nicht, auch die Datei XSUB.COM von der Systemdiskette auf die Diskette zu kopieren, mit der Sie arbeiten.

Angenommen, Sie wollen mit PIP widerholt verschiedene Kopiervorgänge durchführen, ohne dabei jedesmal die Befehle einzeln aufrufen zu müssen. Die folgende Befehlsdatei KOPIE.SUB kopiert mit Hilfe von PIP eine BAK-Textdatei im Bezugslaufwerk in eine TXT-Datei um, die an-schließend über den Bildschirm und über den Drucker ausgegeben werden soll:

```
XSUB
PIP
$1.TXT=$1.BAK
CON:=$1.TXT
LST:=$1.TXT ^A
M
```

In der ersten Zeile steht XSUB, damit die Parameter auch an die aufgerufenen Programme übergeben werden, d.h. in unserem Fall an PIP. Es folgen der Aufruf von PIP und die PIP-Befehlszeilen so, als hätten wir sie von Hand hinter dem Sternchen eingegeben.

Am Schluß wollen wir PIP wieder verlassen und zu CP/M zurückkehren. Normalerweise drücken wir dazu die ENTER-Taste, für die wir in der



Befehlszeile ^AM (Carriage Return) setzen. Würden wir bei der Erstellung der Befehlsdatei in der letzten Zeile lediglich die ENTER-Taste drücken, erhielten wir eine Leerzeile und SUBMIT würde die gesamte Befehlsdatei nicht ausführen. Achten Sie deshalb darauf, daß Sie in einer Befehlsdatei niemals Leerzeilen verwenden.

Ebenso wie ENTER können wir auch andere Steuerzeichen in der Befehlsdatei mit angeben. Dazu dürfen wir aber nicht die CTRL-Taste drücken, sondern schreiben statt dessen einen senkrechten Pfeil. CTRL–Z etwa ist dann als ^AZ einzugeben.

Nun wollen wir die Befehlsdatei ausführen und geben für $1 wieder unsere Textdatei PROBE.TXT an, hier jedoch ohne Extension, da diese bereits in der Befehlsdatei berücksichtigt ist. Der Aufruf mit SUBMIT sieht dann folgendermaßen aus:

```
SUBMIT KOPIE PROBE
```

Selbstverständlich hätten wir auch bei der Verwendung von XSUB mehrere Parameter übergeben können. Auf diese Weise kann man z.B. ED aufrufen und sich die Befehlsdatei selbst verändern lassen, indem einzelne Zeilen gelöscht oder neu eingefügt werden. Beim Debugger DDT, den wir noch kennenlernen werden, könnten wir sogar so weit gehen, daß wir ein Maschinenprogramm (COM-Datei) oder das CP/M-Betriebssystem während der Ausführung der Befehlsdatei verändern. Sie sehen, daß Ihrer Phantasie hierbei kaum Grenzen gesetzt sind!

# 7 Debugging

Die Überschrift dieses Kapitels wird Sie vielleicht überraschen, besonders dann, wenn Sie sich unter Debugging nichts vorstellen können. Debugging ist aber durchaus nichts Ungewöhnliches in der Computerwelt und dient zur Fehlersuche oder -beseitigung.

Debugging kommt von dem englischen Wort "Bug" (Käfer, Wanze, Motte). Sie werden sich jetzt sicher fragen, was eine Fehlersuche mit Insekten zu tun hat. Dafür gibt es eine ganz einfache Erklärung, jedenfalls wenn man der folgenden überlieferten Legende Glauben schenken darf.

In ihrer Anfangszeit waren die Computer gegenüber heute noch riesig groß, so daß sie ganze Räume oder sogar Häuser füllten. Damals enthielten sie keinesfalls die kleinen ICs, wie heute, sondern Relais, die für jede Informationseinheit (Bit) einen Stromkreis öffneten oder schlossen.

Eines Tages trat bei einer solchen Computeranlage eine Störung auf. Als man der Sache auf den Grund ging und die unzähligen Relais auf ihre Funktionstüchtigkeit hin überprüfte, fand man heraus, daß sich in einem Relais eine Motte eingeklemmt hatte, die die ganze Anlage zum Erliegen brachte.

An diesem Tag wurde der Begriff Debugging geboren, der sich seitdem aber nicht nur auf Relais bezieht. Heute ist ein Debugger meist ein Hilfsprogramm, das man einsetzt, wenn in einem Programm "der Wurm drin sitzt" . Mit einem Debugger kann man Speicherbereiche in hexadezimaler Schreibweise auflisten und ändern. Häufig bietet er aber noch weitere Möglichkeiten, wie beispielsweise Speicherbereiche verschieben oder mit einem konstanten Wert füllen.

Auch für CP/M gibt es einen solchen Debugger, der sich unter dem Namen DDT.COM auf Ihrer Systemdiskette befindet. Mit ihm werden wir uns in diesem Kapitel noch eingehend befassen.

## 7.1 DUMP

Die einfachste Möglichkeit, eine Datei in hexadezimaler Schreibweise zu betrachten, bietet das DUMP-Programm. DUMP ist sowohl als COM- als auch als ASM-Datei auf der Systemdiskette enthalten.

Die Extension ASM sagt aus, daß es sich hierbei um einen Assembler-Quelltext handelt, den wir in eine COM-Datei umwandeln können. Bei einem Quelltext haben wir aber die Möglichkeit, ihn nach unseren eigenen Wünschen noch zu verändern. Doch mehr dazu im nächsten Kapitel, in dem wir uns noch eingehend mit der Assembler-Programmierung beschäftigen werden.

Im Augenblick interessiert uns vor allem die COM-Datei, die wir sofort ausführen können und mit der wir jetzt unser Beispiel-Programm PROBE. TXT im Hexcode auflisten wollen.

Wenn Sie nur ein Laufwerk zur Verfügung haben, müssen Sie die DUMP. COM-Datei erst auf die Diskette kopieren, die auch PROBE.TXT enthält. Der Programmaufruf findet dann mit

```
DUMP PROBE.TXT
```

statt, d.h. Sie müssen hinter DUMP noch den Namen der Datei angeben, die Sie listen möchten.

Bei zwei Laufwerken legen Sie am besten die Systemdiskette in Laufwerk A und die Diskette, die PROBE.TXT enthält, in Laufwerk B. Dann starten Sie DUMP mit

```
DUMP B:PROBE.TXT
```

In beiden Fällen erhalten Sie jetzt die nachfolgende Auflistung:

```
0000 44 69 65 73 20 69 73 74 20 5A 65 69 6C 65 20 31 0010 2E OD
OA 44 69 65 73 20 69 73 74 20 5A 65 69 6C 0020 65 20 32 2E OD OA
44 69 65 73 20 69 73 74 20 5A 0030 65 69 6C 65 20 33 2E OD OA 44
69 65 73 20 69 73 0040 74 20 5A 65 69 6C 65 20 34 2E OD OA 44 69
65 73 0050 20 69 73 74 20 5A 65 69 6C 65 20 35 2E OD OA 1A 0060
1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 0070 1A 1A
1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A
```

Wenn Sie bisher noch wenig mit der Innenwelt Ihres CPC beschäftigt haben, werden Ihnen diese Zeichen auch nicht allzuviel sagen. Trotzdem wollen wir sie uns einmal näher ansehen.



PROBE.TXT ist eine reine Textdatei, die im ASCII-Code abgelegt ist. Deshalb bestehen die Zeichen, die wir hier sehen, aus reinem ASCII-Code. Im Anhang dieses Buches finden Sie eine Tabelle, die die Zuordnung der einzelnen Codes zu den entsprechenden Textzeichen angibt.

In jeder Zeile werden insgesamt 16 Bytes bzw. ASCII-Codes dargestellt, wobei der vierstellige Wert am Zeilenanfang die relative Position zum Dateianfang angibt. Somit beginnt die erste Zeile mit 0000H (0 dez.), die zweite mit OOIOH (16 dez.) usw.

Erinnern Sie sich, daß die erste Textzeile in der Datei PROBE.TXT lautete:

```
Dies ist Zeile 1.
```

Lassen Sie uns einmal die ersten vier Bytes in der Datei untersuchen, wobei wir die ASCII-Code-Tabelle im Anhang zu Hilfe nehmen. Bei richtiger Vorgehensweise finden wir dabei folgendes heraus:

| | **Hex** | **Dez.** | **ASCII-Zeichen** |
|---|---|---|---|
| 1. Zeichen | 44 | 68 | D |
| 2. Zeichen | 69 | 105 | |
| 3. Zeichen | 65 | 101 | e |
| 4. Zeichen | 73 | 115 | s |

Wir erhalten tatsächlich das Wort "Dies". Dieses Spiel könnten wir beliebig fortsetzen, um den gesamten Text der Datei lesbar zu machen, worauf wir hier aber verzichten wollen. Dennoch interessieren uns noch einige Sonderzeichen, die in jeder Textdatei immer wieder vorkommen.

Wenn wir den aufgelisteten ASCII-Code zeichenweise untersuchen, fäll uns mehrfach die Zeichenfolge OD OA auf, erstmals an der Position 11H in der zweiten Zeile. Dabei bedeuten

| Hex | **Dez.** | |
|---|---|---|
| OD | 13 | Carriage Return (Wagenrücklauf) |
| OA | 10 | Line-Feed (Zeilenvorschub) |

Mit diesen Zeichen wird eine Textzeile abgeschlossen und eine neue eingeleitet. Wenn der Computer diese Zeichen liest, setzt er den Cursor an den Anfang der nächsten Zeile. Ähnliches geschieht auch beim Drucker, bei dem das Papier um eine Zeile hochgeschoben und der Druckkopf am Zeilenanfang positioniert wird.

Da DUMP nicht die logische, sondern die physikalische Dateilänge liest und anzeigt, werden immer nur ganze Records (128 Bytes) ausgegeben. Dies ist auch in unserem Beispiel der Fall. Dabei fällt uns auf, daß die letzten Bytes des Records nicht belegt sind und alle das Dateiende-Zeichen 1AH (26 dez., CTRL—Z) enthalten.

Vom Arbeiten mit dem Editor wissen wir, daß jeder Text mit diesem Zeichen abgeschlossen werden muß und daß eine Textdatei solange eingelesen wird, bis das Dateiende-Zeichen auftritt.

Theoretisch würde hier ein 1A-Zeichen durchaus genügen; CP/M füllt jedoch die Bytes des letzten Records, die nicht belegt sind, mit diesem Zeichen auf.

Mit DUMP können Sie jedoch nicht nur ASCII-Dateien, sondern auch beliebige andere Dateien im Hexcode auflisten. Versuchen Sie es doch einmal mit einer COM-Datei, z.B. mit PIP.COM. Solche Dateien bestehen zumeist aus einer Mischung aus Maschinen- und ASCII-Code, denn sämtliche auszugebenden Meldungen sind auch hier im ASCII-Code abgelegt. Die nicht belegten Bytes des letzten Records enthalten hier meist 00-Zeichen.

Mit DUMP können Sie den Inhalt von Dateien zwar ansehen, aber leider nicht verändern. Das ist nur mit DDT möglich, wie wir noch sehen wer-den.

## 7.2 DDT

DDT ist ein universelles Debugging-Programm, mit dem man Speicherinhalte auflisten, ändern oder sogar in mnemonischer Form (Assembler-Quelltext) ausgeben kann. DDT ist ein Standard-CP/M-Programm das sich auf der mitgelieferten Systemdiskette befindet.

### 7.2.1 DDT laden

DDT kann, ähnlich wie PIP, auf zweierlei Weise geladen und aktiviert werden. Wir wollen dies wieder anhand unserer Beispieldatei PROBE.TXT ausprobieren. Geben Sie jetzt

```
DDT PROBE.TXT
```



ein, wenn sich DDT auf der gleichen Diskette wie PROBE.TXT befindet oder

```
DDT B:PROBE.TXT
```

wenn DDT in Laufwerk A (z.B. auf der Systemdiskette) und PROBE.TXT in Laufwerk B abgelegt ist.
DDT meldet sich jetzt mit

```
DDT VERS 2.2
NEXT PC 0180
0100
```

Die Datei PROBE.TXT wurde zusammen mit DDT in den Speicher geladen und ist jetzt zwischen Adresse PC (O100H) und NEXT (0180H) abgelegt. Diese beiden Werte sollten wir uns unbedingt merken (am besten notieren!), denn wir benötigen sie später noch, wenn wir die geänderte Datei wieder auf Diskette schreiben.

Als Alternative können wir DDT und Datei auch separat laden, eine Möglichkeit, die besonders Anwender mit einem Laufwerk zu schätzen wissen. Zunächst geben wir

```
DDT
```

ein, worauf sich **DDT mit**

```
VERS 2.2
```

meldet. Der Strich bedeutet, wie bereits im obigen Fall, daß DDT zur Aufnahme von Befehlen bereit ist. Jetzt legen wir die Diskette mit der zu untersuchenden Datei (in unserem Fall wieder PROBE.TXT) ins Lauf-werk und geben dann

```
IPROBE.TXT
```

ein. Achten Sie darauf, daß zwischen dem Befehl I und dem Dateinamen hier kein Leerzeichen steht! Mit

```
R
```

schließlich wird die Datei in den Speicher geladen und es erscheint wieder die Meldung

```
NEXT PC 0180
0100
```

wie im obigen Fall.

Normalerweise legt DDT die Dateien ab Adresse 100H, dem Beginn des TPA ab. Dies ist auch die Anfangsadresse, an die sämtliche COM-Dateien automatisch geladen werden. Auch wir wollen für den Normalfall diese Adresse beibehalten, solange wir mit DDT arbeiten.

Hinter R können wir noch einen Parameter als Versatz angeben, wenn wir eine andere Ladeadresse als 100H wünschen. So lädt beispielsweise

```
R1000
```

die betreffende Datei mit einem Versatz von 1000H, d.h. ab Adresse 0100H+1000H=1100H. Mit dem Versatz sollten wir allerdings möglichst vorsichtig umgehen, denn er funktioniert nur einmal fehlerfrei nach je-dem DDT-Auruf.

### 7.2.2 SAVE

Bevor wir nun näher auf die eigentlichen DDT-Befehlen eingehen, wollen wir uns bereits an dieser Stelle mit dem SAVE-Befehl beschäftigen. SAVE hat eigentlich nichts mit DDT zu tun und ist ein residenter CP/MBefehl, der Dateien auf Diskette schreibt.

Da DDT keine Möglichkeit vorsieht, geänderte Dateien wieder auf Diskette zu schreiben, müssen wir deshalb auf SAVE zurückgreifen. SAVE ist zwar einfach anzuwenden, erfordert jedoch etwas Rechenarbeit, bei der uns keinesfalls Fehler unterlaufen dürfen!

Bevor wir SAVE anwenden, müssen wir DDT zunächst verlassen. Dies geschieht entweder mit einem Warmstart (<CTRL—C>) oder mit dem Befehl GO (Null, nicht Buchstabe 0!). Durch diesen Vorgang bleibt die Datei im Speicher unverändert erhalten.

Jetzt legen Sie die Diskette ins Bezugslaufwerk, auf die Sie die Datei schreiben möchten und führen nochmals mit <CTRL—C> einen Warmstart aus. Dies sollten Sie ohnehin nach jedem Diskettenwechsel tun. Da SAVE ohne Laufwerksangabe arbeitet, müssen Sie mit

```
B:
```



Laufwerk B als Bezugslaufwerk wählen, wenn Sie die Datei in diesem Laufwerk ablegen möchten.
Die allgemeine Aufrufform von SAVE lautet:

```
SAVE Dateigröße/256 Dateiname
```

Dabei müssen wir neben dem Dateinamen auch die Dateigröße in 256 Byte-Einheiten in dezimaler Form angeben müssen, und hier beginnt die lästige Rechnerei!

Derartige Speichereinheiten, die wir sonst nicht bei CP/M vorfinden, entsprechen mit 256 Bytes genau einer Speicherseite (Page). Eine Page umfaßt somit 100H oder genau 16 Zeilen, wenn wir Dateien mit DUMP oder DDT auflisten.

An dieser Stelle benötigen wir wieder die Speicherangaben, die wir uns nach jedem Einladen einer Datei mit DDT notieren sollten. Betrachten wir hierzu einige Beispiele.
Als wir die Datei PROBE.TXT mit DDT geladen haben, erhielten wir

```
NEXT PC
 0180 0100
```

Die ersten beiden Ziffern dieser Hexadezimalzahlen geben die Pages an. Wenn wir nun diese Ziffern voneinander subtrahieren und eins hinzuzählen, erhalten wir die benötigten Pages. In unserem Beispiel sind es somit 01-01+1=1 Page. Wir können folglich die Datei mit

```
SAVE 1 PROBE.TXT
```

wieder auf Diskette zurückschreiben.

Als nächstes Beispiel betrachten wir eine Datei mit dem Namen LAGER. DAT, die beim Laden durch DDT folgende Werte ergibt:

```
NEXT PC
 0800 0100
```

Diese Datei belegt also 08-01+01 = 8 Pages, so daß wir sie mit

```
SAVE 8 LAGER.DAT
```

abspeichern können. Eigentlich wären wir hier auch mit 7 Pages ausgekommen, da die beiden rechten Ziffern 00 lauten.

Was ist nun aber zu tun, wenn nachfolgend aufgeführte Werte auftreten, wie beispielsweise bei der Datei SUPER.HEX?

```
NEXT     PC
2E80   0100
```

Hier wird die Sache schon etwas komlizierter, denn wir müssen zunächst Hexadezimalzahlen in Dezimalzahlen umwandeln. Dabei hilft uns die abgebildete Tabelle:

| Hex. | Dez. |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| A | 10 |
| B | 11 |
| C | 12 |
| D | 13 |
| E | 14 |
| F | 15 |

Wenn wir jetzt aus 2E eine Dezimalzahl bilden, müssen wir die erste Ziffer mit 16 multiplizieren und die zweite dazuzählen, was unter Zuhilfenahme der Tabelle

```
2*16 + 14 = 46
```

ergibt. Die Anzahl der Pages (dezimal!) beträgt somit:

```
46—1+1 = 46
```

Folglich können wir die Datei SUPER.HEX mit

```
SAVE 46 SUPER.HEX
```

abspeichern. Sollten wir uns verrechnen, kann das katastrophale Folgen haben, besonders dann, wenn zu wenig Pages abgespeichert werden. Im Zweifelsfall ist es ratsam, die Datei unter einem anderen Namen abzulegen, sie dann auszutesten und später in ihren richtigen Namen umzubenennen.



Ebenso wie mit DUMP können wir auch mit DDT den Dateiinhalt in hexadezimaler Schreibweise ansehen, allerdings in einer etwas anderen Form. DDT liest nicht die Datei und listet sie gleichzeitig, sondern kann nur den Speicherinhalt listen. Dabei spielt es keine Rolle, ob im Speicher eine Datei abgelegt ist oder nicht.

Wenn wir nun eine Datei ansehen, müssen wir den Speicherbereich ab 100H listen, nämlich ab der Stelle, an die DDT normalerweise Dateien ablegt. Dies wollen wir wieder mit unserer Datei PROBE.TXT ausprobieren.

Zunächst müssen wir DDT und PROBE.TXT nach einer der oben besprochenen Methoden laden. Ist das geschehen, erscheint der Bindestrich zur weiteren Befehlseingabe. Wenn wir jetzt

```
D
```

eingeben, werden die ersten 12 Zeilen mit je 16 Bytes ab Adresse IOOH gelistet:

```
0100 44 69 65 73 20 69 73 74 20 5A 65 69 6C 65 20 31 Dies ist
Zeile 1 0110 2E OD OA 44 69 65 73 20 69 73 74 20 5A 65 69 6C ..
.Dies ist Zeil 0120 65 20 32 2E OD OA 44 69 65 73 20 69 73 74
20 5A e 2...Dies ist Z 0130 65 69 6C 65 20 33 2E OD OA 44 69 65
73 20 69 73 eile 3...Dies is 0140 74 20 5A 65 69 6C 65 20 34 2E
OD OA 44 69 65 73 t Zeile 4...Dies 0150 20 69 73 74 20 5A 65 69
6C 65 20 35 2E OD OA 1A ist Zeile 5....
0160 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A
0170 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A
0180 1A 84 12 13 C3 69 01 D1 2E 00 E9 2A EC 01 22 E7 ....... i
*..".
0190 08 23 22 ED 08 3A EB 01 32 D5 OA 32 EA OF 32 F4 .#"..:..2.
.2..2.
01A0 10 C3 3D 01 5F 1E C9 11 00 00 OE 12 CD 05 00 32 ..=.......2
01B0 5F 1E C9 21 68 1E 70 2B 71 2A 67 1E EB OE 13 CD ..!h.p+q*g
```

Wir erhalten die gleiche Auflistung wie mit DUMP, nur mit anderen Adressenangaben. Zusätzlich werden sämtliche Bytes, soweit möglich, an der rechten Seite noch als ASCII-Zeichen angegeben. Deshalb sehen wir auch sofort, wo Text abgelegt ist und wo nicht. Für alle Zeichen, die keinem listbaren ASCII-Code entsprechen, erscheint ein Punkt.

Da unsere Datei PROBE.TXT nur wenige Zeichen umfaßt, wird auch noch Speicher angezeigt, der nicht mehr zu PROBE.TXT gehört. Aus diesem Grund interessieren uns alle Adressen, die größer als 180H sind, in diesem Fall nicht

Anders ist es, wenn wir eine Datei ansehen wollen, die mehr Speicher-platz belegt. Reichen nämlich die ersten 12 Zeilen nicht aus, geben wir nochmals D ein, worauf die nächsten 12 Zeilen erscheinen, usw.

Wir können auch eine Speicheradresse vorgeben, ab der 12 Zeilen gelistet werden sollen, z.B.:

D2800

Hier erfolgt die Speicherauflistung ab Adresse 2800H. Es besteht auch die Möglichkeit, eine Anfangs- und Endadresse vorzugeben, wie im folgen-den Beispiel:

D1800,1980

In diesem Fall wird der Speicher von Adresse 1800H bis 1980H angezeigt.

Auf diese Weise können wir nicht nur den Speicherbereich ansehen, der Dateien enthalten kann, sondern auch alle anderen Bereiche des gesamten 64K-Speichers. So haben wir beispielsweise die Möglichkeit, auch einen Blick in das BDOS und BIOS zu werfen.

### 7.2.4 **Speicher ändern**

Als nächstes wollen wir unsere Datei PROBE.TXT mit DDT ändern. Am Anfang steht der ASCII-Code für "Dies", den wir jetzt durch den Code für "Hier" ersetzen wollen. Da die Begriffe "Dies" und "Hier" jeweils vier Zeichen umfassen, bereitet uns die Änderung keine Schwierigkeit.

Wir suchen uns zunächst die Codes für "Hier" aus der ASCII-Code-Tabelle heraus und geben dann

5100

ein, da wir den Speicher ab Adresse 100H ändern wollen. Daraufhin er-scheint die jeweilige Adresse mit dem Inhalt der Speicherzelle. Wollen wir den Inhalt belassen, drücken wir die ENTER-Taste, worauf der Inhalt der nächsten Zelle erscheint. Soll der Inhalt jedoch verändert werden, schreiben wir den neuen Code einfach daneben und drücken erst danach die ENTER-Taste. Durch Eingabe eines Punktes kehren wir wieder in den DDT-Befehlsmodus zurück. Wenn Sie nun "Dies" durch "Hier" richtig aus-getauscht haben, muß auf dem Bildschirm folgendes stehen:

```
-5100
0100 44 48
0101 69 0102
65 0103 73 72
0104 20
```



Eigentlich müßten wir hier nur das erste und das letzte Zeichen ändern, da beide Begriffe in der Mitte zufällig "ie" enthalten. Deshalb haben wir das zweite und dritte Zeichen übergangen, indem wir die ENTER-Taste drückten. Es wäre aber nichts passiert, wenn wir die Zeichen trotzdem neu eingegeben hätten.

Jetzt kontrollieren wir, ob wir die Begriffe richtig ausgetauscht haben, indem wir nochmals mit

D100

den Speicher auflisten und uns rechts die Textzeichen ansehen.

Selbstverständlich kann man mit dem S-Befehl nicht nur ASCII-Code, sondern auch Maschinenprogramme ändern, indem man den Maschinencode neu eingibt. Einen solchen Vorgang bezeichnet man als "Patchen", was in etwa mit "Flicken" zu übersetzen ist.

Falls Sie nun die Datei PROBE.TXT in abgeänderter Form abspeichern möchten, müssen Sie zunächst <CTRL—C> oder <GO> eingeben und sie dann mit

SAVE 1 PROBE.TXT auf

Diskette schreiben (s.o.). **7.2.5**

**Assembler/Disassembler**

DDT besitzt einen einfachen Assembler, mit dem man mnemonische Codes zur Maschinenprogrammierung eingeben kann. Er wird mit

**A** (Speicheradresse)

aufgerufen. Mit diesem Assembler wollen wir uns aber nicht näher beschäftigen, da er bei weitem nicht so komfortabel ist wie der Assembler ASM, den wir im nächsten Kapitel noch kennenlernen werden.

Zum Assembler gibt es noch einen einfachen Disassembler, der den Speicherinhalt in mnemonischer Schreibweise auslistet und mit dem L-Befehl aufgerufen wird. Beispielsweise disassembliert die Eingabe

L1200,1300

den Speicher zwischen den Adressen 1200H und 1300H. Wird kein gültiger Opcode vorgefunden, erscheint ein Fragezeichen.

### 7.2.6 **Weitere Möglichkeiten**

Mit den Befehlen, die wir bisher kennengelernt haben, sind die Möglichkeiten von DDT aber noch nicht erschöpft. Deshalb werden wir in diesem Abschnitt noch auf einige weitere Befehle eingehen, die in erster Linie für den Maschinenprogrammierer von Interesse sind.

Manchmal kommt es vor, daß man einen ganzen Speicherbereich mit einem konstanten Wert füllen möchte. Zu diesem Zweck gibt es den F-Befehl, mit dem wir folgenden Versuch machen. Geben Sie einmal ein:

F2000,2100,0

und anschließend

D2000,2100

Sie werden dabei feststellen, daß der Bereich von 2000H bis 2100H mit lauter Nullen gefüllt ist.

Diesen mit Nullen gefüllten Bereich wollen wir verschieben und zwar so, daß er ab Adresse 4000H abgelegt wird. Möglich ist das mit dem M-Befehl, den wir folgendermaßen einsetzen:

M2000,2100,4000

Die ersten beiden Werte geben dabei die ursprüngliche Anfangs- und Endadresse und der dritte Wert gibt die Anfangs-Zieladresse an. Nach Ausführung des Befehls muß der Block mit Nullen auch ab 4000H erscheinen, was Sie ebenfalls mit dem D-Befehl nachprüfen können.

Wenn Sie mit einem Assembler ein Maschinenprogramm geschrieben haben, möchten Sie es sicher auch austesten, was Sie bei DDT mit dem G-Befehl tun können. Auch hierzu ein Beispiel:

G135A,1764

läßt ein Maschinenprogramm ab der Adresse 135A ablaufen. Bei 1764H wird ein Haltepunkt gesetzt, so daß es dort abbricht. Es ist keinesfalls erforderlich, daß sich an dieser Stelle ein Abbruchbefehl (z.B. RET) befindet, denn DDT setzt ihn hier selbst ein.



Soll die Abarbeitung bei der gegenwärtigen Programmzähler-Adresse beginnen, können Sie den ersten Wert auch weglassen, dürfen aber das Komma nicht vergessen. Bei

**G,** 23AA

beginnt die Abarbeitung bei der gegenwärtigen Programmzähler-Adresse und endet bei Adresse 23AAH.

Eine Abwandlung von G ist der T-Befehl, in dem man eine bestimmte Anzahl von Programmschritten vorgibt, bei deren Abarbeitung jeweils die Prozessorregister ausgegeben werden. Darüber hinaus gibt es noch den U-Befehl, der ebenso wie T arbeitet, jedoch keine Registerinhalte ausgibt. T und U beginnen die Programmausführung bei der jeweiligen Programmzähler-Adresse. So führen die Anweisungen

T23

und

U23

jeweils 23H (35 dez.) Programmschritte aus, wobei im ersten Fall auch die Registerinhalte erscheinen.

Schließlich gibt es noch den X-Befehl, mit dem die Prozessorregister angezeigt und geändert werden können.

X

ohne weiteren Zusatz zeigt die Register an, während beispielsweise XA

den Akkumulatorinhalt aufzeigt, für den man, ähnlich wie beim S-Befehl, einen neuen Wert daneben schreiben kann.

# 8 Assemblerprogrammierung

## 8.1 Ein paar Worte vorweg

Dieses Kapitel ist für diejenigen bestimmt, die die Möglichkeiten der Assemblerprogrammierung unter CP/M nutzen möchten. Bevor wir jedoch näher darauf eingehen, wollen wir zunächst einmal klären, was ein Assembler ist und welche Aufgaben er übernimmt.

Grundsätzlich dient ein Assembler zur Generierung eines ausführbaren Maschinencodes. Zwar könnten Sie ein Maschinenprogramm theoretisch auch mit dem S-Befehl von DDT schreiben, indem Sie sich sämtliche Hexcodes selbst zusammensuchen, um diese zunächst im Speicher und dann auf der Diskette abzulegen. DDT eignet sich aber nur zum Aus-bessern einiger weniger Bytes, was man in der Fachsprache als Patchen bezeichnet. Wenn Sie jedoch ganze Programme damit schreiben, wird dies zu einer sehr mühsamen Angelegenheit, die nicht nur eine genaue Kenntnis des internen Maschinencodes erfordert, sondern auch einige Anforderungen in hexadezimalem Kopfrechnen stellt. Sie müssen nämlich sämtliche relativen und absoluten Adressen korrekt berechnen und den Speicher richtig einteilen können.

Zweifellos erhalten Sie, wenn Sie diese Voraussetzungen erfüllen und sonst keinen Fehler machen, ein ebensolches Maschinenprogramm wie mit einem Assembler. Aber warum sollen wir es uns so kompliziert machen, wenn es auch einfacher geht!

Jeder Assembler benötigt, ganz gleich für welchen Zweck und für welchen Mikroprozessor er eingesetzt wird, einen Quelltext mit mnemonischen Befehlen. Dies sind genormte Schlüsselwörter, die der Assembler erkennt und in einen Maschinencode umwandelt. Wenn Sie in BASIC programmieren, geschieht übrigens etwas ganz ähnliches, nur auf einer weitaus höheren Ebene als bei einem Assembler. Dabei werden die BASIC-Schlüsselwörter in einen internen Code umgewandelt, den der

BASIC-Interpreter versteht und mit dem er entsprechende Maschinenroutinen aufruft.

Bekanntlich kann CP/M nur auf solchen Computern betrieben werden, die einen 8080- oder Z80-Mikroprozessor besitzen, was auch für den CPC zutrifft. Zur Enstehungszeit von CP/M gab es nur den 8080-Prozessor, einen Vorgänger vom Z80 mit einem weitaus geringeren Befehlssatz. Bedauerlicherweise verstehen die Assembler für den 8080 einen anderen mnemonischen Code als die für den Z80, obwohl sie einen Maschinencode generieren, der zum Z80 voll kompatibel ist.

Obwohl es den 8080-Prozessor heutzutage gar nicht mehr gibt und alle modernen Computer mit einem Z80-Prozessor ausgerüstet sind, gehört immer noch ein 8080-Assembler zur Standard-Ausrüstung eines jeden CP/M-2.2-Systems. Wegen seines geringeren Befehlssatzes muß man deshalb auf alle Befehle vezichten, die der Z80-Prozessor zusätzlich bietet.

Trotz dieses Handicaps werden wir uns in diesem Kapitel mit dem CP/M-8080-Assembler beschäftigen, der auf Ihrer Systemdiskette mit-geliefert wird. Selbst mit weniger Befehlen können wir immer noch eine ganze Menge anfangen. Übrigens beziehen sich die meisten Assemblerlistings für CP/M in der Fachliteratur ebenfalls auf den 8080-Assembler.

Sollten Sie jedoch in der glücklichen Lage sein, einen Z80-Assembler zu besitzen (für den CPC werden mehrere angeboten), können Sie damit ebenfalls CP/M-Programme schreiben und somit den gesamten Befehlssatz des Z80 ausschöpfen. Dies gilt selbst dann, wenn der Assembler nur unter AMSDOS arbeitet. In diesem Fall müssen Sie lediglich darauf achten, daß das Assemblerprogramm bei Adresse 100H startet und die genormten BDOS- und BIOS-Schnittstellen verwendet.

Dieses Kapitel kann und soll einen vollständigen Assemblerlehrgang nicht ersetzen, denn dazu gibt es genügend andere Literatur auf dem Markt. Wir wollen uns hier aber mit der Bedienung des 8080-Assemblers befassen und in diesem Zusammenhang auch ein kleines CP/M-Programm schreiben und austesten.

## 8.2 Der Quelltext

Wir wissen bereits, daß ein Assembler einen Quelltext benötigt, den er dann in den eigentlichen Maschinen- oder Objektcode übersetzt. Dieser Quelltext unterliegt bestimmten Regeln und besteht ausschließlich aus



listbarem ASCII-Code, der mit jedem Texteditor erstellt werden kann. Der Quelltext muß in einer Datei abgelegt werden, deren Name die Extension ASM enthält, damit sie der Assembler später lesen kann. Wir wer-den jetzt ein kleines Assemblerprogramm schreiben, das Sie nach Ihrem Namen fragt und Ihnen dann herzliche Grüße von CP/M ausrichtet.

Zunächst geben wir den Quelltext für dieses Programm ein und legen ihn dann in der Datei MUSTER.ASM ab. Sie können den Text mit ED oder mit jedem anderen Texteditor, wie z.B. WORDSTAR, erstellen, der reinen ASCII-Code erzeugt. Wir wollen hier jedoch den Editor ED verwenden, der sich auf der Systemdiskette befindet. Den Umgang mit ED hatten wir bereits in Kapitel 4 kennengelernt. Obwohl dieser Editor nicht gerade komfortabel in der Bedienung ist, eignet er sich dennoch zum Schreiben von Quelltextdateien recht gut.

Zunächst rufen wir ED mit dem Dateinamen MUSTER.ASM auf und gehen anschließend sofort in den i-Modus zum Schreiben und Einfügen von Text über:

```
ED MUSTER.ASM
NEW FILE
 . * i
1 :
```

Jetzt geben wir zeilenweise den folgenden Quelltext ein:

```
;Herzliche Gruesse von
CP/M ; - - - - - - - - - -



ORG 100H ; CP/M-
Programmstart BDOS EQU 5 ;
BDOS-Aufruf CCP EQU 0 ;
Warmstart


MVI C,9 ; 1.String
ausgeben LXI D,P1
CALL BDOS


MVI C,10 ;
Texteingabe LXI D,
EINGABE CALL BDOS


MVI C,9 ; 2.String
```

```
MVI D,0 ; $-Zeichen an Ende von Eingabe
LDA EINGABE+1
MOV E,A
LXI H,EINGABE
DAD  D
INX  H
INX H
MVI A,'$'
MOV M,A


MVI C,9
LXI D,
EINGABE+2 CALL
BDOS


JMP CCP ; Warmstart


P1 DB 'Ihren Namen bitte? $'
P2 DB ODH,OAH,OAH,'CP/M gruesst Sie herztich -
$' EINGABE DB 25
DS 26
END
```

Nach Abschluß der Eingabe schreiben wir den Text mit dem E-Befehl auf Diskette.

Bevor wir nun fortfahren noch einige Erläuterungen zum Aufbau des Quelltextes. Beim Schreiben jeder Zeile müssen Sie folgende Reihenfolge einhalten:

<Label> <Befehlswort> <Argumente> ; <Kommentar>

Ein Label ist eine Markierung für eine symbolische Adresse und wird meist als Argument für Sprungbefehle verwendet. Es kann aber auch, wie in unserem Beispiel, die Stelle kennzeichnen, an der ein bestimmter ASCII-Text abgelegt ist oder an der sich ein Puffer befindet. Das Label muß jeweils, falls vorhanden, an erster Stelle in der Zeile aufgeführt sein und darf keinemanderen Begriff mit vorgegebener Bedeutung, wie z.B. den Befehlswörtern, entsprechen. Manchmal werden Labels mit einem Doppelpunkt abgeschlossen, was beim CP/M-Assembler aber nicht unbedingt erforderlich ist.

An nächster Stelle steht das Befehlswort, das durch mindestens ein Leerzeichen vom Label getrennt sein muß. Es entspricht der mnemonischen Schreibweise für den Assembler. Falls das Befehlswort noch Argumente benötigt, sind diese dahinter aufzuführen, ebenfalls im Abstand von mindestens einem Leerzeichen.



Abschließend folgt der Kommentar, der vom Assembler nicht berücksichtigt wird und einen beliebigen Text zur Erläuterung enthalten kann. Er dient lediglich der Ubersicht und ist keinesfalls zwingend. Wird er aber verwendet, so ist er durch ein Semikolon von dem übrigen Text zu trennen.

Am Anfang eines jeden Quelltextes muß außerdem die ORG-Anweisung stehen, die angibt, ab welcher Stelle im Speicher das Programm später geladen und ausgeführt werden soll. In unserem Fall gibt die ORG-Anweisung die Adresse 100H an, an der sämtliche CP/M-Programme starten.

Doch nun zu den eigentlichen Anweisungen des Programms. Hinter ORG sind die Labels mit den festen Adressen für den BDOS-Aufruf und den Warmstart aufgeführt, denen wir hier die Namen BDOS und CCP gegeben haben. Die eigentliche Zuordnung zu diesen Adressen findet durch EQU statt. Da bei einem BDOS-Aufruf immer die Routine in Adresse 5 anzuspringen ist, lautet die Anweisung entsprechend "BDOS EQU 5".

Als erstes gibt das Programm den String "Ihren Namen bitte?" aus, der unter dem Label P1 am Schluß des Programms aufzufinden ist. Dazu verwendet es die BDOS-Routine mit der Nummer 9, die für diesen Zweck vorgesehen ist. Allerdings müssen Strings, wie auch in diesem Fall, mit einem $-Zeichen abschließen, welches das BDOS als Stringende interpretiert.

Zum eigentlichen Aufruf der BDOS-Routine ist das C-Register mit deren Nummer zu laden (hier Nr. 9) und das Doppelregister D mit der Anfangsadresse des Strings, die hier durch das Label P1 gekennzeichnet ist. Schließlich ist noch BDOS in Adresse 5 aufzurufen und der String erscheint auf dem Bildschirm.

Als nächstes erfolgt eine Texteingabe von der Konsole, die normalerweise Ihren Namen enthalten sollte. Auch hierzu gibt es eine BDOS-Routine, die mit der Nummer 10 aufgerufen wird. Zusätzlich ist noch die Adresse des Puffers im D-Register anzugeben, in den die eingegebenen Zeichen abgelegt werden sollen und der hier mit dem Label "EINGABE" gekennzeichnet ist. Der Puffer kann maximal 25 Textzeichen aufnehmen, benötigt aber am Anfang zwei weitere Bytes, in denen die maximale und die wirkliche Länge des eingegebenen Strings abgelegt wird. Deshalb enthält das erste Byte des Puffers die Zahl 25, worauf sich 26 weitere Bytes anschließen (1 Byte für die wirkliche Stringlänge plus 25 Bytes zur Ablage des Strings).

Nach der Texteingabe, die durch <ENTER> abzuschließen ist, wird sogleich ein weiterer String mit dem Inhalt "CP/M gruesst Sie herzlich -"

ausgegeben, der mit dem Label P2 gekennzeichnet ist. Er beginnt mit einem CR- und zwei LF-Zeichen, damit er zwei Zeilen unter der ersten Zeile erscheint.

Nun soll in der gleichen Zeile auch noch der eingegebene Name aus-gegeben werden, den wir direkt aus dem Eingabepuffer herauslesen. Dies ist jedoch nicht ohne weiteres möglich, denn wir müssen ihn am Schluß noch mit einem $-Zeichen versehen. Erst wenn das geschehen ist, können wir ihn wieder mit der BDOS-Routine 9 ausgeben.

Nachdem nun sämtliche Routinen ausgeführt sind, wollen wir wieder zu CP/M zurückkehren und springen die symbolische Adresse CCP an. Dies hat die gleiche Wirkung wie die Eingabe von <CTRL¯C>, nämlich die Ausführung eines Warmstarts.

Abschließend noch ein paar Worte zur Speicherreservierung für die Strings und den Eingabepuffer. Wie Sie sehen, werden konstante Werte immer hinter der DB-Anweisung abgelegt. Wenn es sich um ASCII-Code handelt, können die entsprechenden Textzeichen direkt angegeben wer-den, wobei sie allerdings in einfache Anführungszeichen einzuschließen sind.

Soll nur freier Speicher für bestimmte Zwecke reserviert werden, wie hier für den größten Teil des Eingabepuffers, muß das mit der Anweisung DS ( Anzahl) geschehen.

Jedes Assernblerlisting sollte mit END abschließen, was wir auch hier nicht vergessen wollen.

## 8.3 Quelltext assemblieren

Nachdem wir nun unseren Quelltext (hoffentlich richtig!) in der Datei MUSTER.ASM abgelegt haben, können wir ihn assemblieren. Dazu benötigen Sie den Assembler ASM.COM, der sich auf Ihrer Systemdiskette befindet. Im Gegensatz zu manchen anderen Assemblern erzeugt er jedoch noch nicht den reinen Maschinencode, sondern eine ASCII-Datei im Intel-Hex-Format, die erst später mit LOAD in Objektcode umgewandelt wird. Zusätzlich wird in einer anderen Datei ein Assembler-PRN-Listing erzeugt, das zur eigentlichen Dokumentation des Programms dient.



Wenn Sie mit einem Laufwerk arbeiten, kopieren Sie am besten die Dateien ASM.COM und LOAD.COM auf die Diskette, die den Quelltext enthält. Bei zwei Laufwerken belassen Sie die Systemdiskette wieder in Laufwerk A und die Diskette mit dem Quelltext in Laufwerk B.
Nun rufen Sie den Assembler mit

```
ASM MUSTER
```

bzw.

```
ASM B:MUSTER
```

auf, worauf die Assemblierung stattfindet. Sollten Ihnen bei der Erstellung des Quelltextes Fehler unterlaufen sein, werden die fehlerhaften Zeilen auf dem Bildschirm angezeigt. Sie müssen dann den Quelltext nochmals korrigieren. Ansonsten erscheint lediglich

```
CP/M ASSEMBLER - VER 2.
0 0181
    000H USE
    FACTOR END OF
    ASSEMBLY
```

auf dem Bildschirm, d.h. der Assembler hat keinen Fehler festgestellt.

Auf Ihrer Diskette befinden sich jetzt zwei weitere Dateien, die wir mit TYPE auflisten können. Zunächst einmal die PRN-Datei, die das Assembler-PRN-Listing enthält:

```
TYPE MUSTER.PRN
                ;Herzliche Gruesse von
                CP/M -------------------
                -------


0100            ORG 100H ; CP/M-
Programmstart
0005 =          BDOS EQU 5 ;BDOS-Aufruf
0000 =          CCP EQU 0 ;Warmstart


0100 0E09       MVI C,9 ; 1.String ausgeben
0102 113201     LXI D,P1
0105 CD0500     CALL BDOS


0108 OEOA       MVI C,10 ; Texteingabe
010A 116601     LXI D,EINGABE
010D CD0500     CALL BDOS
```

```
0110 0E09     MVI C,9 ; 2.String ausgeben
0112 114601   LXI D,P2
0115 CD0500   CALL BDOS

0118 1600     MVI D,0 ; $-Zeichen an Ende von Eingabe
011A 3A6701   LDA EINGABE+1
011D 5F       MOV E,A
011E 216601   LXI H,EINGABE
0121 19       DAD D
0122 23       INX H
0123 23       INX H
0124 3E24     MVI A,'$'
0126 77       MOV M,A

0127 0E09     MVI C,9
0129 116801   LXI D,EINGABE+2
012C CD0500   CALL BDOS

012F C30000   JMP CCP ; Warmstart

0132 496872656EP1 DB 'Ihren Namen bitte? $'
0146 0D0A0A4350P2 DB 0DH2OAH2OAH,'CP/M gruesst Sie herzlich      $'
0166 19       EINGABE DB 25
0167          DS 26
0181          END
```

Wir erhalten also nochmals den Quelltext, der jetzt aber mit Adressen und Maschinencode versehen ist. Wenn wir dagegen mit

```
TYPE MUSTER.HEX
```

die Datei im Intel-Hex-Format listen, erhalten wir folgendes Ergebnis:

```
:100100000E09113201CD05000EOA116601CD050
060                                     :
:100110000E09114601CD050016003A67015F216
600                                     :
:10012000011923233E24770E09116801CD0500C
370
:100130000000496872656E204E616D656E20626
9CF
:100140007474653F20240D0A0A43502F4D20677
2B6
:10015000756573737420536965206865727A6C6
97C :070160006368202D20241923
:0000000000
```

Mit diesem Code können wir allerdings nicht viel anfangen; wir benötigen ihn aber noch zur Erstellung einer ausführbaren COM-Datei. Dies erbzw.



```
LOAD B:MUSTER
```

Nachdem nun LOAD seine Arbeit verrichtet hat, erscheint die Ausgabe:

```
FIRST ADDRESS 0100    (Startadresse)
LAST  ADDRESS 0166    (Endadresse)
BYTES READ    0067    (Gelesene Bytes = Länge der Datei)
RECORDS WRITTEN 01    (Belegte Records)
```

Wenn wir uns jetzt das Directory anschauen, finden wir unsere Datei MUSTER mit fünf verschiedenen Extensionen vor:

MUSTER. **ASM** (Quelltext)
MUSTER. BAK (BAK-Datei des Editors)
MUSTER. PRN (Assembler-PRN-Listing)
MUSTER.HEX (Intel-Hex-Listing)
MUSTER. COM (Ausführbare CP/M-COM-Datei)

Die zuletzt aufgeführte COM-Datei ist das Endresultat unserer Arbeit. Wir wollen unser Programmm auch gleich einmal testen und starten es dazu mit:

```
MUSTER
```

Wenn alles stimmt, erhalten Sie folgende Ausgabe auf dem Bildschirm:

```
Ihren Namen bitte? Franz Huber      (Name eingeben)
CP/M gruesst Sie herzlich - Franz Huber
```

# 9 Hinter den Kulissen

In den vorangehenden Kapiteln haben wir bereits sämtliche Befehle kennengelernt, die wir zum normalen Arbeiten mit CP/M 2.2 benötigen. Normalerweise ist dies völlig ausreichend, sofern Sie nicht in das "Innere" von CP/M einsteigen möchten; aber genau das wollen wir in diesem Kapitel tun. Vielleicht gehören Sie zu den begeisterten Assemblerprogrammierern, die eigene CP/M-Programme schreiben und das Letzte aus dem CP/M-Betriebssystem herausholen möchten. Dies setzt allerdings eine genaue Kenntnis der Speicherorganisation, der Aufrufadressen und einiger wichtiger Parameter voraus, mit denen wir uns nachfolgend beschäftigen werden.

An dieser Stelle sei aber nochmals betont, daß das vorliegende Buch keinen Assemblerlehrgang enthält, denn dieser würde selbst ein ganzes Buch füllen. Sie finden hier jedoch alle wichtigen Systeminformationen, die Sie für Ihre eigenen CP/M-Maschinenprogramme auf dem CPC benötigen.

## 9.1 Interne Speicherorganisation

Bereits seit dem ersten Kapitel wissen wir, daß das CP/M-Betriebssystem aus den drei Grundelementen CCP, BDOS und BIOS besteht und daß der CPC einen Arbeitspeicher (TPA) von ca. 40K enthält. CCP und BDOS sind geräteunabhängig und bei jedem CP/M-2.2-Computer identisch, während das BIOS die Schnittstelle zwischen CP/M und der Hardware darstellt und für jeden Computertyp separat erstellt werden muß.

Das CP/M-Betriebssystem ist auf den beiden äußeren Spuren der im Systemformat formatierten Disketten abgelegt. Im Gegensatz zu den meisten anderen Computern ist das BIOS beim CPC nicht auf Diskette, sondern im Floppy-ROM gespeichert und somit ständig resident. Genauer gesagt wird es größtenteils auch von AMSDOS mitverwendet, da dort die interne Dateiverwaltung mit der von CP/M nahezu identisch ist.

Doch zunächst sehen wir uns einmal die Standard-Speicheraufteilung beim CPC unter CP/M an:

```
0000H - OOFFH  Grundseite (Zeropage)
0100H - 96FFH  TPA
9700H - 9EFFH  CCP
9FOOH - ACFFH  BDOS
ADOOH          BIOS-Sprungtabelle (ruft Routinen im Floppy-ROM auf)
```

Die höheren RAM-Adressen bis BFFFH werden vom BIOS als Datenpuffer und zur Ablage von anderen Parametern genutzt.

Die Grundseite (0000H-OOFFH) ist ebenfalls genormt und setzt sich folgendermaßen zusammen:

```
0000H - 0002H JMP-Vektor für Warmstart
0003H         IOBYTE für Gerätezuordnungen einzelner Kanäle
0004H         Bezugslaufwerk und
Benutzernummer 0005H - 0007H JMP-Vektor für
BDOS-Aufrufe
0008H - 005BH Für RST-Aufrufe und teilweise
ungenutzt 005CH - 007FH Standard-File-Control-
Block (FCB) 0080H - OOFFH Standard-DMA-Puffer (
Datenpuffer)
```

Für eigene Anwendungen sind hier die JMP-Vektoren für den Warmstart und die BDOS-Aufrufe am wichtigsten, denn sie kommen in den meisten CP/M-Programmen häufig vor.

Für den Normalfall sollte die Speicheranordnung für CCP, BDOS und die BIOS-Sprungtabelle beibehalten werden; es besteht jedoch die Möglichkeit, das Betriebssystem auf andere Speichergrößen anzupassen. Dies geschieht durch den Aufruf des Programms MOVCPM, das allerdings beim CPC mit großer Vorsicht anzuwenden ist. Wird nämlich der Versuch unternommen, den größtmöglichen RAM-Speicher für CP/M zu nutzen, wird dies beim CPC nicht funktionieren, da wichtige vom BIOS benutze RAM-Bereiche dabei überschrieben werden.

Es bleibt die Frage, ob es überhaupt sinnvoll ist, CP/M auf einen kleineren Speicher anzupassen. Da der TPA beim CPC nicht übermäßig groß ist, besteht hierzu eigentlich kein Anlaß, obwohl CP/M bereits auf einem System mit 20K lauffähig ist. Viele CP/M-Programme benötigen sogar mehr Speicher, als der CPC bietet, so daß man gegebenenfalls auf eine Speichererweiterung zurückgreifen muß.

Der Vollständigkeit halber betrachten wir hier ein Beispiel, das CP/M auf eine Speichergröße von 24K anpaßt. Da diese bei MOVCPM, ähnlich wie bei SAVE, in 256-Byte-Einheiten anzugeben ist, müssen wir sie zunächst umrechnen und erhalten dafür 96 Einheiten. Dann wird



**MOVCPM 96** *

aufgerufen, wodurch das angepaßte Betriebssystem ab Adresse 900H abgelegt wird. Dies ist übrigens auch die gleiche Adresse, an der SYSGEN die eingelesenen Systemspuren ablegt, wenn es ohne Zusatz aufgerufen wird. Jetzt erscheint die Meldung:

```
CONSTRUCTING 24k CP/M vers
2.2 READY FOR "SYSGEN" OR "
SAVE 34 CPM24.00M"
```

Sie haben daraufhin die Möglichkeit, das Betriebssystem entweder **mit**

**SYSGEN ***

direkt in die Systemspuren zu schreiben oder es zunächst mit SAVE in einer gewöhnlichen Datei (z.B. CPM24.COM) abzulegen, falls Sie es noch ändern möchten. Geben Sie

**SYSGEN CPM24.COM**

ein, wird CP/M aus der Datei gelesen und in die Systemspuren übertragen.

## 9.2 Directory und File-Control-Block (FCB)

Auf Diskette abgelegte Dateien müssen nach einem ganz bestimmten Schema verwaltet werden, wofür das Directory zuständig ist. Es enthält nämlich weit mehr Informationen, als mit dem DIR-Befehl aufgelistet werden können.

Das Directory befindet sich bei CP/M immer in der Spur, die den Systemspuren unmittelbar folgt, beim CPC also in Spur 2, wenn das Betriebssystem die Spuren 0 und 1 belegt. In dieser Spur sind die ersten vier Sektoren zu je 512 Bytes, die insgesamt zwei Blöcke aufnehmen können, für das Directory reserviert. Es kann somit maximal 64 Einträge zu je 32 Bytes aufnehmen, einschließlich derjenigen, die ein zusätzliches Extent für eine Datei darstellen.

Wir wollen uns als nächstes den Aufbau eines Directory-Eintrags anhand der folgenden Tabelle einmal genauer ansehen:

| Von Byte | Bis Byte | Inhalt |
|---|---|---|
| 0 | | Benutzernummer (0 bis 31) |
| 1 | 8 | Dateiname im ASCII-Code (Großbuchstaben) |
| 9 | 11 | Extension im ASCII-Code (Großbuchstaben) |
| 12 | | Nummer des Eintrags (Extent) |
| 13 | 14 | Für interne Zwecke reserviert |
| 15 | | Anzahl der abgelegten Records im Extent (0 bis 128) |
| 16 | 31 | Nummern der für die Datei belegten Blöcke, wobei maximal 16 Blöcke pro Extent verwaltet werden können. |

Insgesamt kann ein Directory-Eintrag also 16K verwalten, sofern ein Block 1K umfaßt. Dies ist beim CPC und den meisten anderen CP/MSystemen der Fall. Enthält eine Datei jedoch mehr als 16K, so ist für jede weiteren angefangenen 16K jeweils ein Eintrag in das Directory vorzunehmen. In diesem Fall spricht man von Zusatzeinträgen oder Extents.

Der File-Control-Block (FCB) befindet sich nicht auf der Diskette, sondern im Arbeitsspeicher, wo er einen Directory-Eintrag zur Bearbeitung aufnimmt. Der FCB hat folgenden Aufbau:

| **Von Byte** | **Bis Byte** | **Inhalt** |
|---|---|---|
| 0 | | Laufwerksnummer (0 für Bezugslaufwerk, 1 bis 16 für Laufwerk **A** bis P) |
| 1 | 8 | Dateiname im ASCII-Code (Großbuchstaben) |
| 9 | 11 | Extension im ASCII-Code (Großbuchstaben) |
| 12 | | Nummer des Eintrags (Extent) |
| 13 | 14 | Für interne Zwecke reserviert |
| 15 | | Anzahl der abgelegten Records im Extent (0 bis 128) |
| 16 | 31 | Nummern der für die Datei belegten Blöcke, wobei maximal 16 Blöcke pro Extent verwaltet werden können. |
| **32** | | Nummer des nächsten Records |
| **33** | **34** | Nummer des absoluten Records bei Direktzugriff |
| **35** | | Flag für Überlauf bei BDOS-Funktion **35** |

Sie sehen also, daß das Directory und der FCB nahezu identisch aufgebaut sind, wobei der FCB jedoch noch einige Zusatzinformationen enthält. So ist hier statt der Benutzernummer das aktuelle Laufwerk angegeben, und die Bytes 32 bis 35 enthalten Informationen, die insbesondere zum Lesen und Schreiben von relativen Dateien benötigt werden.

Die Bytes 1 bis 8, die für den Dateinamen reserviert sind, enthalten normalerweise den 7-Bit-Standard-ASCII-Code, jedoch hat auch das achte Bit bei einigen Zeichen eine bestimmte Bedeutung. So dient es beispielsweise zur Kennzeichnung eines Dateiattributes, das einen Schreibschutz



oder eine Systemdatei kennzeichnet. Andere höchstwertige Bits dagegen haben keine Bedeutung und stehen dem Benutzer für eigene Anwendungen zur Verfügung.

## 9.3 BDOS-Aufrufe

Das BDOS enthält insgesamt 40 Funktionen, die in eigenen CP/M-Programmen verwendet werden können. Im Kapitel 8 hatten wir bereits ein kleines Assemblerprogramm entwickelt, das einen String von der Tastatur einliest und auf dem Bildschirm ausgibt. Dies war jedoch nur ein kleiner Vorgeschmack auf die vielen Aufgabenstellungen, die mit Hilfe der BDOS-Aufrufe gelöst werden können.

Für jeden BDOS-Aufruf ist die angegebene Nummer im C-Register abzulegen und dann Adresse 5 aufzurufen. Nachfolgend sind sämtliche Aufrufe aufgeführt, einschließlich der Register zur Übergabe bzw. Übernahme der Parameter. 8-Bit-Werte werden im E-Register und 16-Bit-Werte im Doppelregister DE an die BDOS-Routinen übergeben. Dagegen erfolgt die Übernahme von den Routinen mit dem A-Register für 8-Bit-Werte und mit dem Doppelregister HL für 16-Bit-Werte.

0 **SYSTEM RESET:**

Führt einen Warmstart aus. Keine Parameter sind notwendig.

1 **CONSOLE INPUT:**

Liest ein Zeichen von der Konsole in A ein und gibt es auf dem Bildschirm (Echofunktion) aus.

2 CONSOLE **OUTPUT:**

Gibt ein Zeichen in E auf der Konsole (Bildschirm) aus. Dabei werden Vorgaben wie Tabulator, <CTRL–S> oder <CTRL–P> (zusätzliche Druckerausgabe) berücksichtigt.

3 **READER INPUT:**

Liest ein Zeichen in A vom Lochstreifenleser oder von einem anderen zugeordneten Gerät (z.B. Modem) ein.

4 PUNCH OUTPUT:

Gibt ein Zeichen in E auf den Lochstreifenstanzer oder ein auf anderes zugeordnetes Gerät (z.B. Modem) aus.

5 LIST OUTPUT:

Gibt ein Zeichen in E auf dem Drucker aus.

6 DIRECT CONSOLE I/O:

Unmittelbare Konsoleneingabe eines Zeichens in A oder Ausgabe in E; sollte normalerweise nicht verwendet werden.

7 GET **IOBYTE:**

Fragt den Inhalt des IOBYTES (A) ab.

8 SET **IOBYTE:**

Schreibt einen Wert in E in das IOBYTE.

9 PRINT STRING:

Gibt einen String (Adresse in DE) auf dem Bildschirm aus, der mit einem S-Zeichen abgeschlossen sein muß.

10 READ CONSOLE BUFFER:

Ubernimmt einen String von der Konsole und legt ihn in einem Puffer ( Adresse in DE) ab. Der String ist mit CR oder LF abzuschließen.

11 GET CONSOLE STATUS:



**12 RETURN VERSION NUMBER:**

Liefert die Nummer (HL) der verwendeten CP/M-Version. H liefert 0 für CP/M 80 und L die eigentliche Nummer (22H für CP/M 2.2).

**13 RESET DISK SYSTEM:**

Setzt das Diskettensystem zurück (keine Parameter). Dabei wird immer Laufwerk A angewählt und die Datenpufferadresse (DMA) auf 80H gesetzt.

**14 SELECT DISK:**

Bestimmt ein Laufwerk (E) zum Bezugslaufwerk. Für Laufwerk A ist der Wert 0, für Laufwerk B der Wert 1 usw. zu übergeben.

**15 OPEN FILE:**

Öffnet Datei (DE auf FCB-Adresse).

**16 CLOSE FILE:**

Schließt Datei (DE auf FCB-Adresse).

**17 SEARCH FOR FIRST:**

Sucht den ersten Eintrag einer Datei im Directory (DE auf FCB-Adresse). Dabei sind auch mehrdeutige Dateinamen zulässig.

**18 SEARCH FOR NEXT:**

Sucht nach Ausführung von Funktion 17 den nächsten Eintrag (keine Parameter erforderlich).

**19 DELETE FILE:**

Löscht Datei (DE auf FCB-Adresse), sofern sie nicht schreibgeschützt ist. Bei mehrdeutigen Dateinamen können auch mehrere Dateien gelöscht werden.

20 **READ SEQUENTIAL:**

Liest den nächsten fortlaufenden Record einer Datei (DE auf FCBAdresse) und legt ihn im Datenpuffer (DMA) ab. Der Recordzähler wird dabei um eins erhöht.

**21 WRITE SEQUENTIAL:**

Schreibt den nächsten fortlaufenden Record einer Datei (DE auf FCB-Adresse) aus dem Datenpuffer (DMA) in eine Datei. Der Recordzähler wird dabei um eins erhöht.

22 **MAKE FILE:**

Legt eine Datei neu an (DE auf FCB-Adresse), ähnlich wie bei OPEN. In das Directory erfolgt aber zunächst ein Leereintrag, da die Datei noch keine Daten enthält.

**23 RENAME FILE:**

Nennt Dateien um. Dazu ist ein spezieller FCB erforderlich (Adresse in DE), in dem die ersten 16 Bytes den alten und die restlichen 16 Bytes den neuen Dateinamen enthalten.

**24 RETURN LOGIN VECTOR:**

Fragt ab, welche Laufwerke angeschlossen sind und gibt in HL einen 16-Bit-Wert zurück. Das niederwertigste Bit steht dabei für Laufwerk A und das höchstwertigste Bit für Laufwerk P. Das jeweilige Bit ist gesetzt, wenn das betreffende Laufwerk in Betrieb ist, anderenfalls ist es gelöscht.

**25 RETURN CURRENT DISK:**

Gibt das Bezugslaufwerk an. Dabei wird der Wert 0 für Laufwerk A, 1 für Laufwerk B usw. in A übermittelt.

**26 SET DMA ADDRESS:**

Setzt Datenpufferadresse (in DE) für Schreib- und Lesevorgänge.



27 **GET ADDR (ALLOC):**

Gibt die Adresse des Blockbelegungsverzeichnisses in HL an. Diese Funktion wird z.B. von STAT verwendet, um den freien Diskettenspeicher zu ermitteln.

28 **WRITE PROTECT DISK:**

Versieht das selektierte Laufwerk mit einem temporären Schreibschutz ( keine Parameter).

29 **GET READ-ONLY VECTOR:**

Ermittelt schreibgeschützte Laufwerke, indem in HL ein 16-Bit-Wert zurückgegeben wird (ähnlich wie bei Funktion 24).

30 **SET FILE ATTRIBUTES:**

Setzt Dateiattribute für System- oder schreibgeschützte Dateien bzw. setzt sie zurück, indem der entsprechende Eintrag vom FCB (Adresse in DE) in das Directory übertragen wird.

**31 GET DPB ADRRESS:**

Ermittelt die Adresse des Disk-Parameter-Blocks (in HL).

**32 SET/GET USER CODE:**

Dient zur Ermittlung und zum Setzen der aktiven Benutzernummer. Soll die Nummer ermittelt werden, enthält E den Wert 255, die dann in A zurückgegeben wird. Beim Setzen enthält E die gewünschte Benutzernummer.

33 **READ RANDOM:**

Dient zum Lesen eines Records für Direktzugriffe. Die Recordnummer wird hierbei aber nicht wie beim sequentiellen Lesen (Funktion 20), sondern durch Erweiterungseinträge im FCB (Adresse in DE) übergeben.

34 **WRITE RANDOM:**

Dient zum Schreiben eines Records, wobei die Recordnummer durch Erweiterungseinträge im FCB (Adresse in DE) übergeben wird.

35 **COMPUTE** FILE SIZE:

Ermittlung der Dateigröße in Records. Durch Vorgabe der FCB-Adresse in DE wird die Größe berechnet und in den Erweiterungsbytes für relativen Dateizugriff des FCB abgelegt.

**36 SET RANDOM RECORD:**

Ermittlung der augenblicklichen Recordnummer einer Datei und Ablage in den Erweiterungsbytes für relativen Dateizugriff des FCB (Anfangsadresse in **DE).**

**37 RESET DRIVE:**

Zurücksetzen einzelner Laufwerke durch Übergabe eines 16-Bit-Parameters in DE, wobei das niederwertigste Bit Laufwerk A entspricht.

**38 ACCESS DRIVE:**

beim CPC ohne Bedeutung.

**39 FREE DRIVE:**

Beim CPC ohne Bedeutung.

**40 WRITE RANDOM WITH ZERO:**

Schreiben eines Records wie bei Funktion 34, wobei der Record jedoch zuvor mit Nullen gefüllt wird.

## 9.4 Mehr über das BIOS

Das BIOS ist der einzige geräteabhängige Teil des CP/M-Systems; er führt Funktionen auf einer weit niedrigeren Ebene als das BDOS aus. Es beginnt mit einer genormten Sprungtabelle, deren Anfangsadresse genau um OEOOH über dem BDOS-Anfang liegt. Auch beim CPC ist diese Sprungtabelle vorhanden, sie enthält allerdings Adressen, die sich auf das Floppy-ROM beziehen.



Hier nun die BIOS-Adressen, wie sie bei einem normalen CP/M-Start erzeugt werden, d.h. wenn das Betriebssystem nicht mit MOVCPM verschoben wurde:

| | |
|---|---|
| ADOOH | Kaltstart |
| ADO3H | Warmstart |
| ADO6H | Konsolenstatusabfrage (A=0 wenn keine Taste gedückt, sonst **A=255)** |
| ADO9H | Eingabe eines Zeichens von der Konsole in **A** |
| ADOCH | Ausgabe eines Zeichnes in C an die Konsole |
| ADOFH | Ausgabe eines Zeichens in C an den Drucker |
| AD12H | Ausgabe eines Zeichens in C an den Lochstreifenstanzer |
| AD15H | Eingabe eines Zeichens in A vom Lochstreifenleser |
| AD18H | Schreib-/Lesekopf der Floppy auf Spur 0 stellen |
| AD1BH | Laufwerk anwählen (Nummer 0 bis 15 in C für Laufwerk **A** bis P) |
| AD1EH | Spur anwählen (Nummer in BC) |
| AD21H | Record anwählen (Nummer in BC) |
| AD24H | Adresse (in BC) für Recordpuffer (DMA) setzen |
| AD27H | Record lesen, muß vorher angewählt sein |
| AD2AH | Record schreiben, muß vorher angewählt sein |
| AD2DH | Abfrage des Druckerstatus |
| AD3CH | Recordnummer (in BC) anhand von Skewing-Tabelle ( Beginn in DE) übersetzen (Ergebnis in HL) |

Außer dieser Sprungtabelle enthält das BIOS für jedes Laufwerk noch einen Disk-Parameter-Block (DPB) und einen Disk-Parameter-Header (DBH). Die Anfangsadressen beider Tabellen können beim CPC folgen-dermaßen abgefragt werden:

| | |
|---|---|
| BE40/41H | Zeiger auf DPH für Laufwerk A (AE58H) |
| BE41/42H | Zeiger auf DPB für Laufwerk A (ADD8H) |

Falls das Betriebssystem nicht verschoben wurde, beginnt also der DPH bei Adresse AE58H und der DPB bei Adresse ADD8H. Diese beiden Tabellen wollen wir uns einmal genauer ansehen. Die darin enthaltenen CPC-spezifischen Werte sind für beide Laufwerke in Klammern gesetzt.

Beginnen wir mit der DPH-Tabelle (für Laufwerk A ab AE58H, für B ab AE68H):

| Von Byte | Bis Byte | Inhalt |
|---|---|---|
| 0 | 1 | Adresse der Skewing-Übersetzungstabelle (0, da beim CPC nicht verwendet) |
| 2 | 7 | Zwischenspeicher für BDOS, enthält aktuelle Spur-, Sektor- und Recordnummer |
| 8 | 9 | Adresse des Directory-Puffers (für **A** u. B = AE78H) |
| 10 | 11 | Adresse des DPB (A=ADD8H, B=AE18H) |
| 12 | 13 | Prüfsummentabelle für Directory (A=ADF1H, B=AE31H) |
| 14 | 15 | Adresse der Blockbelegungstabelle (A=AEO1H, B=AE41H) |

Die Skewing-Übersetzungstabelle enthält die Reihenfolge der Sektoren, in welcher der Zugriff in jeder Spur auf der Diskette stattfindet. Manchmal ist es aus Zeitgründen günstiger, die Aufzeichnung nicht mit dem nächst folgenden Sektor fortzusetzen, sondern zwei oder drei Sektoren zu überspringen.

Hier ein Beispiel: Angenommen, eine Diskette mit 20 Sektoren benötigt für eine Umdrehung 20 ms, d.h. 1 ms, um von einem zum nächsten Sektor zu gelangen. Nachdem nun der Inhalt eines Sektors gelesen wurde, muß er erst in den Speicher übertragen werden, was z.B. 2.5 ms dauert. Ist eine Datei in fortlaufenden Sektoren abgelegt, müßte in diesem Fall die Diskette eine ganze Umdrehung durchmachen, um auf den nächsten Sektor zugreifen zu können, denn nach 1 ms ist der Datentransfer noch nicht abgeschlossen. Werden jedoch jeweils zwei Sektoren ausgelassen, kann nicht erst nach 20, sondern bereits nach 3 ms mit dem Zugriff auf den nächsten Sektor begonnen werden, was insgesamt eine wesentlich schnellere Datenübertragung ermöglicht. Die Skewing-Übersetzungstabelle könnte in diesem Fall folgendermaßen aussehen:

1, 4, 7, 10, 13, 16, 19,
2, 5, 8, 11, 14, 17, 20,
**3, 6, 9,** 12, 15, 18

Obwohl jeweils zwei Sektoren ausgelassen werden, wird dennoch die gesamte Spur genutzt, denn sie kann jetzt anstelle von 20 Umdrehungen bereits während 3 Umdrehungen vollständig gelesen bzw. beschrieben werden.

Die CPC-Floppy enthält allerdings keine Skewing-Tabelle, wie der DPH-Tabelle zu entnehmen ist. Es besteht aber noch die Möglichkeit, die Sektornummern gleich in der Form der Skewing-Tabelle zu formatieren, was dieselbe Wirkung zur Folge hätte. Ob die Floppy allerdings davon Gebrauch macht, ist nicht bekannt, jedenfalls arbeitet sie ohnehin schon recht schnell.



Als weiteres taucht der Begriff Blockbelegungstabelle auf. Dabei handelt es sich um ein Verzeichnis, das für jeden Block auf der Diskette ein Bit enthält. Ist ein Block belegt, so ist auch das entsprechende Bit in der Tabelle gesetzt, anderenfalls ist es gelöscht.

Das Directory enthält noch eine Prüfsummentabelle, mit deren Hilfe CP/M feststellt, ob ein Diskettenwechsel ohne <CTRL—C> vorgenommen wurde. In diesem Fall stimmt die Prüfsumme nicht mehr, was einen BDOS-Error zur Folge hat.

Kommen wir jetzt zum Disk-Parameter-Block, der verschiedene Angaben über die physikalische Struktur der Diskette enthält. Für Laufwerk A erscheint er ab Adresse ADD8H und für Laufwerk B ab Adresse AE18H. Der Inhalt beider Tabellen ist dabei vollkommen identisch. Hier nun der DPB, wobei die Werte für den CPC wieder in Klammern stehen:

| Von Byte | Bis Byte | Inhalt |
|---|---|---|
| 0 | 1 | Records pro Spur (36) |
| 2 | | Blockverschiebefaktor (3) |
| 3 | | Blockmaske (7) |
| 4 | | Extent-Maske (0) |
| 5 | 6 | Max. Blockanzahl -1 (170) |
| 7 | 8 | Max. Anzahl der Directory-Einträge (63) |
| 9 | 10 | Allocation 0 und 1 (C000H) |
| 11 | 12 | Zu überprüfende Blöcke bei Diskettenwechsel (16) |
| 12 | **14** | Anzahl der Systemspuren (2) |

Nachfolgend eine kurze Erläuterung zu den einzelnen Parametern. Der Blockverschiebefaktor gibt die Zweierpotenz der Anzahl der Records pro Block wieder. Sein Wert ist hier gleich 3, da

$2^A3$ * 128 (Recordgröße) = 1024 (Blockgröße)

ergibt. Aus diesem Wert errechnet sich auch die Blockmaske, in der die 3 ( Verschiebefaktor) niederwertigsten Bits gesetzt sind:

0000 0111 binär = 7

Allocation 0 und 1 gibt die Anzahl der Directory-Blöcke an, wobei für jeden Block ein Bit gesetzt ist. In diesem Fall steht aber das niederwertigste Bit links und das höchstwertigste rechts. Da der CPC vier solcher Blöcke enthält, erhält man

1111 0000 0000 0000 binär = 0000H

Eine weitere wichtige Aufgabe des BIOS besteht darin, die logischen Records zu je 128 Bytes an der richtigen Stelle der physikalischen Sek-

toren (beim CPC 512 Bytes) unterzubringen bzw. zu lesen. Diesen Vorgang bezeichnet man in der Fachsprache als Blocking bzw. Deblocking.

## 9.5 Anpassen von CP/M-Software

Abschließend wollen wir noch einen kurzen Blick auf die Anpassung von Software richten. Dabei interessiert uns jetzt nicht die Software, die bereits fertig für den CPC geliefert wird und nur noch einer geringen Anpassung bedarf, sondern die Programme, die Sie von einem anderen CP/M-System auf den CPC übertragen.

Ist eine direkte Übertragung wegen unterschiedlicher Diskettenaufzeichnungsformate nicht möglich, bietet sich in erster Linie eine Verbindung über die RS232- oder V24-Schnittstelle an. In viele andere Computer ist sie bereits eingebaut, in den CPC jedoch nicht. Es gibt allerdings verschiedene Firmen, die eine solche Schnittstelle zusammen mit einem passenden Terminalprogramm anbieten, wobei die Datenübertragung meist über den RDR:- und PUN:-Kanal stattfindet. Da es aber eine Vielzahl von Terminalprogrammen gibt, wollen wir hier auf eine genaue Beschreibung verzichten, die dem Programm bzw. der Schnittstelle ja ohnehin beiliegt. Sollte ein Terminalprogramm nur unter AMSDOS arbeiten, so ist dies nicht hinderlich, denn wir können auch unter AMSDOS CP/M-Programme auf Diskette speichern.

Ist ein CP/M-Programm als COM-Datei auf Diskette abgelegt, ist es häufig schon lauffähig, ohne daß eine weitere Anpassung nötig ist. Dabei darf das betreffende Programm aber nicht mehr Arbeitsspeicher benötigen, als im CPC vorhanden ist.

Anpassungen sind zumeist bei den Programmen erforderlich, die spezielle Bildschirm- oder Druckersteuerzeichen verwenden. So verwendet WORDSTAR beispielsweise Steuerzeichen zur Cursorsteuerung und zum Löschen des Bildschirms, ohne die das Programm nicht richtig funktioniert. Im Handbuch des CPC bzw. Ihres Druckers können Sie aber sämtliche benötigten Steuerzeichen nachschlagen. Viele CP/M-Programme werden mit einem speziellen Installationsprogramm geliefert, mit dessen Hilfe man diese Steuerzeichen eingeben kann. Zumindest liegt aber eine genaue Installationsanweisung mit Patch-Adressen bei, in die man mit Hilfe von DDT die entsprechenden Werte einschreiben kann.

Echte Schwierigkeiten dürften nur bei solchen Programmen auftreten, die andere Adressen als die Standard-BDOS- und BIOS-Sprungadressen verwenden. In diesem Fall handelt es sich allerdings nicht mehr um reine CP/M-Software, sondern um ein gerätespezifisches Programm, das nur für einen speziellen Computer geschrieben wurde.

# 10 CP/M-Befehlsübersicht

In diesem Kapitel finden Sie eine Zusammenfassung sämtlicher CP/M 2.2 Befehle und Dienstprogramme, in alphabetischer Reihenfolge geordnet. Zusätzlich aufgenommen wurden außerdem alle Schneider-CPC-spezifischen Programme, die sich auf der mitgelieferten Systemdiskette befinden, sofern sie unter CP/M lauffähig sind.

Im Gegensatz zur Beschreibung in den vorangehenden Kapiteln sind hier die Unterbefehle (z.B. für ED und PIP) mit aufgeführt und großenteils mit Beispielen versehen. Diese Aufteilung verleiht dem Kapitel den Charakter eines ausführlichen Nachschlagewerkes, aus dem sich besonders der fortgeschrittene CP/M-Anwender die benötigten Informationen schnell beschaffen kann.

## AMSDOS

Gerätespezifischer CPC-Befehl (Datei AMSDOS.COM)

Beschreibung:

AMSDOS schaltet den CPC-Computer vom CP/M-Modus in den BASIC/AMSDOS-Modus zurück, wobei ein Reset durchgeführt wird. Sämtliche Daten im Speicher gehen dabei verloren.

Aufruf:

```
AMSDOS
```



## ASM

Nichtresidenter Standard-CP/M 2.2-Befehl (Datei ASM.COM)

Beschreibung:

ASM ist ein 8080-Assembler, der Quelltexte übersetzt, die in einer Datei mit der Extension ASM abgelegt sind. Die Übersetzung findet dabei nicht direkt in den ausführbaren Maschinen- oder Objektcode statt, sondern zunächst in den Intel-Hex-Code, der als ASCII-Text in einer Datei mit der Extension HEX abgelegt wird. Dieser Code wird erst mit LOAD (s.u.) in eine ausführbare COM-Datei umgewandelt.

Neben der Datei im Intel-Hex-Format erzeugt ASM noch eine weitere Datei mit der Extension PRN, welche das komplette Assemblerlisting zur Dokumentation enthält.

Fehlerhafte Quelltextzeilen werden nicht mit assembliert und auf dem Bildschirm ausgegeben.

Aufruf:

```
ASM (Laufwerk:) ASM-Datei
```

Die ASM-Datei darf nur mit Einfachnamen, also ohne die Extension ASM angegeben werden.

Beispiel:

```
ASM MUSTER
```

übersetzt die Quelltextdatei MUSTER.ASM und erzeugt die Datei MUSTER.ASM im Intel-Hex-Format sowie die Datei MUSTER.PRN, die das Assembler-PRN-Listing enthält.

Optionen:

An den Dateinamen kann hinter einem zusätzlichen Punkt noch eine dreistellige Option angefügt werden (nicht zu verwechseln mit Extension!). Dabei bedeuten

1. Zeichen: Quellaufwerk der ASM-Datei
2. Zeichen: Ziellaufwerk für die HEX-Datei
3. Zeichen: Ziellaufwerk für die PRN-Datei

Wird anstelle des Laufwerks der Buchstabe Z angegeben, so wird die betreffende Datei nicht erzeugt. Steht an dritter Stelle ein X, so wird die PRN-Datei nicht auf Diskette

abgelegt, sondern statt dessen auf dem Bildschirm ausgegeben.

Beispiel:

```
ASM PROBE.BAX
```

übersetzt die Datei PROBE.ASM in Laufwerk B und erzeugt eine HEX-Datei in Laufwerk A. Das Assembler-PRN-Listing wird nicht gespeichert, sondern erscheint auf dem Bildschirm.



## BOOTGEN

Gerätespezifischer CPC-Befehl (Datei BOOTGEN.COM)

Beschreibung:

BOOTGEN liest den CP/M-BOOT-Sektor (Spur 0, Sektor 1) von einer Diskette, die diesen Sektor enthält und kopiert ihn auf eine andere Diskette. Der BOOT-Sektor wird benötigt, um CP/M von BASIC/AMSDOS aus zu starten.

BOOTGEN arbeitet nur mit Laufwerk A, weshalb zuerst die Quell- und dann die Zieldiskette eingelegt werden muß.

Aufruf:

```
BOOTGEN
```

Nach dem Aufruf ist zunächst die Quelldiskette mit dem BOOT-Sektor und dann die Zieldiskette in Laufwerk A einzulegen.

## CHKDISC

Gerätespezifischer CPC-Befehl (Datei CHKDISC.COM)

Beschreibung:

Überprüft, ob die Disketteninhalte in Laufwerk A und B identisch sind (Verify). CHKDISC sollte immer nach dem Kopieren ganzer Disketten mit COPYDISC (s.u.) eingesetzt werden.

Aufruf:

```
CHKDISC
```

Es folgt die Aufforderung, die zu vergleichenden Disketten in Laufwerk A und Laufwerk B einzulegen. Alle nicht übereinstimmenden Sektoren werden angezeigt.



## CLOAD

Systemspezifischer CPC-Befehl (Datei CLOAD.COM)

Beschreibung:

CLOAD liest eine Datei von Kassette und speichert sie auf Diskette. CLOAD wird meist zur Übertragung von ASCII-Dateien eingesetzt.

Aufruf:

```
CLOAD "Kassettendatei" Diskettendatei
```

Der Name der Kassettendatei muß dabei in Anführungszeichen eingeschlossen sein.

Beispiel:

```
CLOAD "BRIEFE" BRIEF.TXT
```

Hier wird die Kassettendatei "BRIEFE" in die Diskettendatei BRIEF.TXT übertragen.

## COPYDISC

Systemspezifischer CPC-Befehl (COPYDISC.COM)

Beschreibung:

Kopiert den Disketteninhalt von Laufwerk A nach Laufwerk B, wobei die Zieldiskette gegebenenfalls formatiert wird.

Aufruf:

```
COPYDISC
```

Bevor der Kopiervorgang stattfindet, erfolgt die Aufforderung, die Quelldiskette in Laufwerk A und die Zieldiskette in Laufwerk B einzulegen.



## CSAVE

Systemspezifischer CPC-Befehl (Datei CSAVE.COM)

Beschreibung:

CSAVE liest eine Datei von Diskette und speichert sie auf Kassette. CSAVE wird meist zur Übertragung von ASCII-Dateien eingesetzt.

Aufruf:

```
CSAVE Diskettendatei "Kassettendatei"
```

Der Name der Kassettendatei muß dabei in Anführungszeichen eingeschlossen sein.

Beispiel:

```
CSAVE BRIEF.TXT "BRIEFE"
```

Hier wird die Diskettendatei BRIEF.TXT in die Kassettendatei "BRIEFE" übertragen.

# DDT

Nichtresidenter Standard-CP/M 2.2-Befehl (Datei DDT.COM)

Beschreibung:

DDT ist ein universeller Debugger, um Speicherinhalte und Dateien zu untersuchen und gegebenenfalls zu ändern.Soll eine Datei bearbeitet werden, wird sie normalerweise ab der Adresse 100H abgelegt, an der sonst auch CP/M-Programme geladen und abgearbeitet werden. In diesem Fall untersucht man den Speicherbereich, in dem sich die Datei befindet.

DDT eignet sich weniger zum Schreiben ganzer Maschinenprogramme als vielmehr zum Andern einzelner Speicherzellen. Dieser Vorgang, den man auch als Patchen bezeichnet, muß häufig zur Anpassung von CP/M-Software durchgeführt werden, wenn das mitgelieferte Installationsprogramm keine Anpassung an den CPC vor-sieht.

Anwendungs-
beispiele:

Speicherinhalt ansehen
Speicherinhalt ändern
Ausführen von Maschinenprogrammen, wobei Haltepunkte ( Breakpoints) gesetzt werden können
Speicherbereiche verschieben
Speicherbereiche mit konstantem Wert füllen
Assemblieren von 8080-Quelltext (nur bedingt möglich)
Disassemblieren von Maschinencode (nur bedingt möglich)

Aufruf:

```
DDT (Laufwerk:) Dateiname
```

oder

```
DDT
```

Im ersten Fall wird die zu untersuchende Datei gleich mitgeladen, im zweiten Fall nicht. In beiden Fällen erscheint dann ein Bindestrich, der zur Eingabe spezieller DDT-Befehle auffordert:

```
— (DDT—Befehl)
```



Achtung! Eine geänderte Datei kann nicht mit DDT wieder auf Diskette geschrieben werden. Dazu ist zunächst DDT entweder mit
GO oder

```
<CTRL—C>
```

(Warmstart) zu verlassen und die Datei dann mit SAVE (s.u.) abzuspeichern.
Beispiel zum DDT-Aufruf mit Dateinamen:

```
DDT COPY.COM
```

lädt DDT und gleichzeitig die Datei COPY.COM in den Speicher ab Adresse 100H. DDT befindet sich anschließend im Befehlsmodus, so daß weitere Befehle zur Bearbeitung der Datei erteilt werden können.

## DDT-Befehle:

A

Dieser Befehl ruft einen einfachen 8080-Assembler auf und ist in der Form von

```
A Anfangsadresse (hex.)
```

einzugeben. Daraufhin erscheint am linken Rand die Anfangsadresse und fordert zur Eingabe eines mnemonischen 8080-Befehlswortes auf. Handelt es sich dabei um einen gültigen Befehl, wird er durch Drücken der ENTER-Taste sogleich übersetzt und der Maschinencode an der richtigen Stelle im Speicher abgelegt. Sodann erscheint die nächstfolgende Adresse, worauf ein neues Befehlswort einzugeben ist, usw. Ist das Befehlswort ungültig, erscheint ein Fragezeichen und fordert zur erneuten Eingabe auf.

Der Assembler ist durch einfaches Drücken der ENTER-Taste wieder zu verlassen. Eine Verwendung von symboli-

schen Adressen und Namen (Labels) ist nicht möglich und der Maschinencode ist nicht verschiebbar (relokatibel).

Beispiel:

```
A3000
```

ruft den Assembler auf, worauf mit der Eingabe von Befehlswörtern begonnen werden kann. Der übersetzte Maschinencode wird ab Adresse 3000H abgelegt.

D

zeigt einen Speicherinhalt auf dem Bildschirm an. Er erscheint als hexadezimales Listing, wobei darstellbare ASCII-Zeichen rechts im Klartext stehen. Der Aufruf von D kann auf verschiedene Arten stattfinden:

D

(ohne Adressenangabe) listet die nächsten 12 Zeilen, von denen jede 16 Bytes darstellt. Wurde DDT gerade aufgerufen (mit oder ohne Dateiname) beginnt die Darstellung automatisch bei 100H. Durch erneute Eingabe von D werden dann jeweils die nächsten 12 Zeilen aus-gegeben.

```
D Adresse
```

arbeitet genauso wie D, beginnt aber bei vorgegebener Anfangsadresse.

Beispiel:

```
D3000
```

gibt 12 Zeilen ab Adresse 3000H aus.

Darüberhinaus kann D auch eine Anfangs- und Endadresse enthalten:

```
D Anfangsadresse, Endadresse
```

So listet

```
D1800,2000
```

den Speicherinhalt zwischen den Adressen 1800H und 200011 auf.



F

Füllt einen Speicherbereich mit einem vorgegebenen konstanten Wert. Der Aufruf findet folgendermaßen statt:

```
F Anfangsadresse, Endadresse, Wert
```

Beispiel:

```
F3000,3FFF,AA
```

füllt den Speicher zwischen den Adressen 3000H und 3FFFH ( je einschließlich) mit dem konstanten Wert AAH.

G

Führt ein Maschinenprogramm aus, wobei die Möglichkeit besteht, bis zu zwei Haltepunkte (Breakpoints) vorzugeben. G wird folgendermaßen aufgerufen:

```
G Anfangsadr., Breakpointl, Breakpoint2
```

Fehlt die Anfangsadresse, beginnt die Ausführung beim momentanen Programmzählerstand. Auch ist die Angabe eines zweiten Breakpoints nicht unbedingt erforderlich.

Achtung! Beim Setzen der Haltepunkte ist unbedingt da-rauf zu achten, daß sie immer auf das erste Byte eines Maschinenbefehls zeigen, da ansonsten das Programm abstürzen kann.

Beispiele:

```
G124A,1271,1344
```

beginnt die Programmausführung bei 124AH, unterbricht sie bei 1271H und beendet sie bei 1344H.

```
G, 327F
```

führt ein Maschinenprogramm ab der gegenwärtigen Programmzähleradresse bis Adresse 327FH aus. Da keine Anfangsadresse angegeben ist, muß hinter G unbedingt ein Komma stehen.

**H**

berechnet gleichzeitig Summe und Differenz zweier Hexadezimalwerte:

Beispiele: I

**H ZAHL** 1, ZAHL 2

Es erscheint als Ausgabe:

```
Summe Differenz
```

```
H1000,800
1800 0800

H100,200
0300 FF00
```

Dient zur Eingabe eines Dateinamens, falls dies beim Aufruf von DDT noch nicht geschehen:

```
I Dateiname
```

Der Dateiname wird im File-Control-Block (FCB) ab Adresse 5DH abgelegt, so daß die Datei dann mit dem R-Befehl (s.u.) geladen werden kann.

Beispiel:

**ILAGER.DAT**

legt den Dateinamen LAGER.DAT **im** FCB ab.

L

**L ist** ein einfacher Disassembler, der 8080-Quellcode erzeugt. Er wird mit

```
L Anfangsadresse, Endadresse
```

aufgerufen. Fehlt die Anfangsadresse, wird die Disassemblierung bei der gegenwärtigen Programmzähleradresse begonnen, fehlt die Endadresse, werden insgesamt nur 11 Zeilen mit Quelltext aufgelistet.

Beispiel:



```
L34A7,361F
```

Disassembliert den Maschinencode zwischen den Adressen 34A7H und 361 FH.

Verschiebt einen Speicherbereich und wird folgendermaßen aufgerufen:

```
M Anfangsadresse(alt), Endadresse(alt),
Anfangsadresse(neu)
```

Beispiel:

```
M1000,1FFF,3000
```

verschiebt den Speicherbereich zwischen 1000H und IFFFH ( je einschließlich) und legt ihn ab der Adresse 3000H ab.

R

Liest nach dem Aufruf von DDT eine Datei von Diskette in den Speicher. Vorher muß jedoch der Dateiname mit dem I-Befehl (s.o.) eingegeben und im FCB abgelegt sein.

```
R
```

(ohne Zusatz) lädt dann die Datei und speichert sie ab Adresse 100H.

Hinter R kann auch noch angegeben werden, mit welchem Versatz die Datei geladen werden soll:

```
R Versatz
```

Beispiel:

Eine Datei, deren Name bereits mit dem I-Befehl erfaßt wurde, soll nicht wie gewöhnlich ab Adresse 100H, sondern ab Adresse 6000H abgelegt werden. Der Versatz beträgt demnach:

```
 6000H
-0100H
 5F00H
```

Die Datei ist folglich mit dem Befehl:

R5F00

zu laden.

S

Mit dem S-Befehl kann der Speicherinhalt byteweise aufgelistet und geändert werden. Der Aufruf findet folgendermaßen statt:
S Anfangsadresse

Auf dem Bildschirm erscheint jetzt die Anfangsadresse und der Inhalt dieser Speicherzelle. Rechts davon ist der Cursor positioniert und wartet auf Eingabe.

Soll der Inhalt der Speicherzelle unverändert bleiben, ist lediglich die ENTER-Taste zu drücken. Ansonsten ist der neue Wert neben den alten zu schreiben und ebenfalls die ENTER-Taste zu drücken.

In beiden Fällen erscheint daraufhin die nächste Adresse, und der gleiche Vorgang beginnt von neuem. Durch die Eingabe eines Punktes bricht der S-Befehl ab und kehrt in den DDT-Befehlsmodus zurück.

Beispiel: S1234

listet den Speicherinhalt byteweise ab Adresse 1234H und bietet die Möglichkeit zur Anderung.

T

arbeitet ein Maschinenprogramm in Einzelschritten ab.. Dabei werden nach jedem Schritt die Prozessorregisterinhalte angezeigt. T wird in Form von:
T Anzahl

aufgerufen, wobei die Anzahl der abzuarbeitenden Programmschritte vorzugeben ist. Die Abarbeitung beginnt bei der gegenwärtigen Programmzähleradresse.



Fehlt die Anzahl der Schritte, so wird nur ein Programmschritt ausgeführt.

Beispiel: T25

arbeitet 25H (37 dez.) Programmschritte ab, beginnend bei der augenblicklichen Programmzähleradresse, die gegebenenfalls mit dem X-Befehl (s.u.) einzustellen ist.

U

Schrittweise Abarbeitung von Maschinenprogrammen wie bei T, jedoch werden die Registerinhalte nur im Anschluß daran angezeigt.

Beispiel: U10

führt, beginnend bei der augenblicklichen Programmzähleradresse, 10H (16 dez.) Einzelschritte aus und zeigt erst nach dem 16. Schritt die Prozessorregisterinhalte an.

X

Dient zum Anzeigen und Ändern der Prozessorregisterinhalte einschließlich Flags und wird folgendermaßen aufgerufen:
X Register oder Flag

Daraufhin erscheint der Inhalt des betreffenden Registers bzw. Flags. Soll er unverändert bleiben, ist lediglich die ENTER-Taste zu drücken, ansonsten ist ein neuer Wert einzugeben und ebenfalls die ENTER-Taste zu drücken.
Folgende Register bzw. Flags können aufgerufen werden:

| | |
|---|---|
| A | Akkumulator |
| B | Doppelregister BC |
| D | Doppelregister DE |
| H | Doppelregister HL |
| S | Stackpointer |
| P | Programmzähler |
| C | Carry-Flag |

| | |
|---|---|
| Z | Zero-Flag |
| M | Minus-Flag |
| E | Paritätsflag |
| I | Interdigit-Carry-Flag |

Beispiele:

XB

zeigt das Doppelregister BC an, dessen Inhalt dann wahlweise verändert werden kann oder nicht. Das gleiche gilt bei
XC

für das Carry-Flag, für das allerdings nur die Werte 0 (nicht gesetzt) oder 1 (gesetzt) in Frage kommen.



## DIR

Residenter Standard-CP/M 2.2 Befehl

Beschreibung:

Mit dem DIR-Befehl wird das gesamte Directory ( Inhaltsverzeichnis) einer Diskette oder ein Teil davon aufgelistet. Die Dateinamen erscheinen dabei in der Reihenfolge ihrer Ablage, d.h. nicht alphabetisch sortiert.

Aufruf:

```
DIR (Laufwerk:)
```

oder

```
DIR (Laufwerk:) Dateiname
```

Dabei kann der Dateiname ein- oder mehrdeutig sein.

Beispiele:

**DIR**

listet das Directory des Bezugslaufwerkes und

**DIR B:**

das Directory von Laufwerk B.

```
DIR PROBE.TXT
```

sucht die Datei PROBE.TXT. Ist sie auf der Diskette vorhanden, wird der Name PROBE.TXT ausgegeben, wenn nicht, erscheint der Hinweis:

```
NO FILE
```

Der Befehl

```
DIR *.COM
```

listet sämtliche COM-Dateien und

```
DIR B*.*
```

diejenigen Dateinamen, die mit dem Buchstaben B beginnen.

# DISCCHK

Gerätespezifischer CPC-Befehl (Datei DISCCHK.COM)

Beschreibung:

Überprüft, ob die Inhalte zweier Disketten identisch sind ( Verify), wozu beide Disketten abwechselnd in Laufwerk A einzulegen sind. DISCCHK sollte immer nach dem Kopieren ganzer Disketten mit einem Laufwerk (s. DISCCOPY) eingesetzt werden.

Aufruf:

DISCCHK

Es folgt die Aufforderung, beide Disketten abwechselnd in Laufwerk A einzulegen. Sämtliche nicht übereinstimmenden Sektoren werden angezeigt.



# DISCCOPY

Gerätespezifischer CPC-Befehl (Datei DISCCOPY.COM)

Beschreibung:

Kopiert den gesamten Inhalt einer Diskette auf eine andere, wobei die Zieldiskette gegebenenfalls formatiert wird. DISCCOPY arbeitet nur mit einem Laufwerk, wes-halb die Disketten während des Kopiervorgangs mehrfach gewechselt werden müssen.

Aufruf:

DISCCOPY

Es folgt die Aufforderung, abwechselnd Quell- und Ziel-diskette in Laufwerk A einzulegen.

## DISCKIT2

Gerätespezifischer CPC-Befehl (Datei DISCKIT2.COM)

Beschreibung:

DISCKIT2 ist ein komfortables Dienstprogramm, das Disketten mit einem oder zwei Laufwerken kopiert, formatiert oder verifiziert. DISCKIT2 wird nur für den CPC6128 geliefert, ist aber auch auf dem CPC 464 und 664 lauffähig.

Die Bedienung ist menügesteuert, wobei die jeweiligen Menüpunkte mit den Funktionstasten des Zifferntastenblocks auszuwählen sind. Weitere Informationen zu DISCKIT2 befinden sich in Kapitel 2.6.

Aufruf:

```
DISCKIT2
```

DISCKIT2 stellt automatisch fest, ob ein oder zwei Laufwerke angeschlossen sind. Bei einem Laufwerk ist zum Kopieren und Verifizieren ein mehrfacher Diskettenwechsel nötig, ähnlich wie beispielsweise bei DISCCOPY.



## DUMP

Residenter CP/M 2.2-Befehl

Beschreibung:

DUMP liest Dateien von Diskette und listet sie in hexadezimaler Form auf dem Bildschirm.

Aufruf:

```
DUMP (Laufwerk:) Dateiname
```

Beispiel:

```
DUMP MUSTER.COM
```

listet die Datei MUSTER.COM im Hexcode.

## ED

Nichtresidenter Standard-CP/M 2.2-Befehl (Datei ED.COM)

Beschreibung:

ED ist ein Texteditorprogramm, das sich zum Erstellen und Bearbeiten von Textdateien jeglicher Art eignet. Dabei werden die Texte in einen Textpuffer geschrieben oder müssen nach dem Aufruf von ED dorthin von Diskette übertragen werden.

Bei ED handelt es sich um einen einfachen Editor mit geringem Bedienungskomfort, der weit hinter demjenigen von professionellen Texverarbeitungsprogrammen, wie z.B. WORDSTAR, zurücksteht.

Anwendungs-beispiele:

Texte schreiben
Textzeilen löschen
Textzeilen einfügen
Begriffe im Text suchen und austauschen

Aufruf:

```
ED (Laufwerk:) Dateiname
```

Dabei ist der Name der Datei anzugeben, die neu angelegt oder bearbeitet werden soll. Nach dem Aufruf erscheint:

```
NEW FILE
t *     (Befehl)
```

wenn die Datei neu angelegt wird oder nur

```
:*      (Befehl)
```

wenn die Datei bereits vorhanden ist und weiter bearbeitet werden soll. Hinter dem Doppelpunkt und dem Sternchen sind die ED-Befehle einzugeben.

Beispiele:

```
ED PROBE.TXT
```

ruft die Datei PROBE.TXT auf oder legt sie neu an. Mit

```
ED B:BRIEF.TXT
```



geschieht das gleich mit der Datei BRIEF.TXT, jedoch in Laufwerk B.

## ED-Befehle:

Achtung! Bei den nachfolgend aufgeführten Befehlen tritt häufig der Wert n auf. Für n steht dabei die jeweilige Anzahl von Zeichen, Zeilen usw. Für n kann man aber auch ein Ziffernkreuz (#) angegeben, das für die größtmögliche Anzahl von n (max. *65535)* steht. ED berücksichtigt dabei immer die Gesamtzahl von Zeichen oder Zeilen in einer Datei oder im Puffer.

Ist für einen Befehl +/-n angegeben, so kann bei positiven Werten das Pluszeichen auch entfallen.

n:

Setzt den Textzeiger an den Anfang der n-ten Zeile.

Beispiel:

```
25:
```

Der Zeiger steht jetzt am Anfang der *25.* Zeile im Textpuffer.

nA

Dieser Befehl liest n Zeilen aus der Textdatei in den Textpuffer zur Bearbeitung ein.

Beispiele:

30A

liest *30* Zeilen in den Puffer und #A

liest sämtliche Zeilen der Textdatei. Der letzte Befehl sollte immer bei nicht zu umfangreichen Dateien Anwendung finden, wenn sie einschließlich möglicher Erweiterungen in den Puffer hineinpassen.

**+/-B**

Setzt den Textzeiger an den Pufferanfang (B oder +B) oder an das Pufferende (¯B).

**+/-nC**

Beispiele: Setzt den Textzeiger von seiner augenblicklichen Position +n Zeichen vor oder -n Zeichen zurück.

-20C
Der Textzeiger wird um 20 Zeichen zurückgesetzt.

**+/-nD**

154C

Der Textzeiger wird um 154 Zeichen vorgerückt.

Beispiele:

Löscht von der augenblicklichen Position des Textzeigers die +n folgenden oder die -n davorliegenden Zeichen.

Angenommen, der Pufferzeiger steht auf dem 1000. Zeichen, dann werden mit
20D

die Zeichen 1000 bis 1019 und mit

-12D

die Zeichen 988 bis 999 gelöscht.

E

Beendigung der Textverarbeitung und Rückkehr zu CP/M. Der Text wird aus dem Puffer in eine Zwischendatei ( Extension $$$) geschrieben, diese dann in den beim Aufruf von ED angegebenen Dateinamen umbenannt und eine eventuell vorher vorhandene Datei in eine BAK-Datei umbenannt.

Beispiel:



Die Bearbeitung der Textdatei BRIEF.TXT, die sich schon vor dem ED-Aufruf auf Diskette befand, soll abgeschlossen werden. Spätestens bei der Ausführung des E-Befehls wird eine Zwischendatei BRIEF.$$$ angelegt, in die der Pufferinhalt abgelegt wird. Im Anschluß daran wird die ursprüngliche Datei BRIEF.TXT in BRIEF.BAK und die Zwischendatei BRIEF.BAK in BRIEF.TXT umbenannt.

nF"Begriff"

Dieser ED-Befehl sucht das n-te Auftreten des vorgegebenen "Begriffs" ab der augenblicklichen Textzeigerposition. Wird für n keine Zahl angegeben, wird n=1 angesetzt. Der hinter n stehende Buchstabe "f" sollte normalerweise klein geschrieben werden, da bei einem großen F der "Begriff" in Großschrift umgewandelt und in dieser Form nach ihm gesucht wird.

Sollen dem Suchbefehl noch weitere Befehle folgen, muß dieser zusätzlich mit einem CTRL–Z abgeschlossen werden.

Wurde der gesuchte "Begriff" gefunden, steht der Textzeiger auf dem dem "Begriff" folgenden Zeichen.

Beispiele:

3fHurra!

sucht das dritte Auftreten des Begriffs "Hurra!". Lautet der Befehl dagegen:
FHurra!

sucht ED nach dem ersten Begriff in der Form von "HURRA! ".

H

H schließt die Textbearbeitung ab. Dieser Befehl hat die gleiche Funktion wie E, kehrt jedoch nicht zu CP/M zu-rück, sondern ruft die bearbeitete Textdatei erneut auf, so daß sie weiterbearbeitet werden kann.

I

Dieser Befehl schaltet den Einfügemodus ein. Zunächst ist der Textzeiger auf die einzufügende Zeile zu stellen und dann der I-Befehl zu erteilen, worauf neue Zeilen eingefügt werden können.

Ist I großgeschrieben, werden alle eingefügten Texte in Großschrift umgewandelt, bei kleinem i ist Groß- und Kleinschrift möglich.

Beispiele:

In einen Text sollen zwischen den derzeitigen Zeilen 4 und 5 weitere Zeilen eingefügt werden. Dazu ist zunächst der Textzeiger auf Zeile 5 zu setzen und dann i einzugeben:

```
5:
i
```

Die erste eingefügte Zeile ist somit die neue Zeile 5, während die Nummern der alten Zeilen, die 5 oder größer waren, um eins nach oben rücken. Dieser Vorgang wieder-holt sich bei jeder zusätzlich eingefügten Zeile.

Da i klein angegeben wurde, können die eingefügten Zeilen in Groß- und Kleinschrift geschrieben werden.

```
-B
 i
```

In diesem Fall wird der Zeiger an das Textende gesetzt, so daß neuer Text angefügt werden kann.

```
i"Begriff"^Z
```

Dieser I-Befehl fügt den "Begriff" an der Position des Textzeigers ein. Auch hier gilt, daß bei einem großen I sämtliche Zeichen des einzufügenden Begriffs in Großschrift umgewandelt werden, während sonst ein kleines i angegeben werden muß.

Beispiel:

```
iTasse^Z
```

fügt das Wort "Tasse" an der Textzeigerposition ein.

```
ni"Begriffl"^Z"Begriff2"^Z"Begriff3"^Z
```



Zunächst wird "Begriff 1" gesucht, "Begriff2" an "Begriffl" angehängt und alle Zeichen bis zum Beginn von "Begriff3" gelöscht.

Beispiel:

```
5iRose^ZGeranie^ZTulpe^Z
```

Nach dem 5. Auftreten des Wortes "Rose" wird an dieses " Geranie" angefügt und sämtliche Zeichen zwischen "Rose" und "Tulpe" gelöscht.

+/-nK

Beispiele:

Dieser Befehl löscht n Zeilen vor bzw. hinter dem Textzeiger.

Angenommen, der Textzeiger steht am Anfang von Zeile 12, dann werden im Falle von

```
-3K
```

die Zeilen 9, 10, 11 und im Falle von

```
4K
```

die Zeilen 12, 13, 14 und 15 gelöscht.

+/-nL

verschiebt den Textzeiger um n Zeilen vor bzw. zurück.

Beispiele:

```
3L
```

setzt den Zeiger um 3 Zeilen vor und

```
-5L
```

setzt ihn um 5 Zeilen zurück.

nM"Befehl"^Z

führt den angegebenen ED-"Befehl" n mal hintereinander aus. Ist n gleich 0 oder 1, wird der "Befehl" so lange

wiederholt ausgeführt, bis die Datei zu Ende ist oder ein Fehler auftritt.

Beispiel: 5M1K^Z

löscht fünfmal hintereinander die nächste Zeile und ersetzt somit den Befehl 5K.

nN"Begriff"^A Z

sucht das n-te Auftreten von "Begriff". Funktioniert wie der F-Befehl mit der Ausnahme, daß bei Nichtauffinden von " Begriff" weiterer Text aus der Datei in den Textpuffer nachgeladen und ebenfalls durchsucht wird.

0

Abbruch der Textbearbeitung bei gleichzeitigem Rücksprung in die Originaldatei, wobei keine Änderungen übernommen werden. Zunächst erfolgt jedoch noch die Sicherheitsabfrage: 0– (Y/N) ?

Sie ist mit Y (Ja) oder N (Nein) zu beantworten.

**+/-nP**

Dieser Befehl zeigt n Textblöcke zu je 24 Zeilen auf dem Bildschirm an, und zwar vor oder hinter der aktuellen Textzeigerposition. Ist n gleich 0, beginnt die Ausgabe mit der Zeile, auf die der Textzeiger gerade zeigt.

**Q**

Abbruch der Textbearbeitung und Rückkehr zu CP/M. Im Gegensatz zum E-Befehl wird die bearbeitete Datei aber nicht gespeichert.



R"Dateiname"

Dient zum Einfügen eines Textblocks, der aus einer Datei mit der Extension LIB in den Textspeicher gelesen wird. Der Textzeiger steht anschließend am Ende des eingefügten Textblocks. Statt dieser Datei kann auch die Zwischendatei X$$$$$$$.LIB gelesen werden.

Beispiel:

```
REINFUEG.LIB
```

liest einen Textblock, der in der Datei EINFUEG.LIB abgelegt ist, in den Textpuffer.

**nS"Begriff 1"^Z"Begriff 2"^Z**

Sucht und ersetzt n mal den "Begriff 1" durch den "Begriff 2". Ist für n kein Wert angegeben, findet dieser Vorgang nur einmal statt.

Ist S groß geschrieben, findet die Suche und der Austausch nur in Großbuchstaben statt, bei kleinem s dagegen in Groß- und Kleinschrift.

Beispiel:

```
sApfel^ZBirne^Z
```

sucht ab der gegenwärtigen Textzeigerposition das Wort " Apfel" und ersetzt dieses durch "Birne".

**+/-nT**

Listet n Zeilen vor bzw. hinter der aktuellen Textzeiger-position.

Beispiele:

Angenommen, der Textzeiger zeigt auf Zeile 20. Durch 4T

werden die Zeilen 20 bis 24 und durch

```
-3T
```

die Zeilen 18 bis 20 gelistet.

+/-U

+/-V
Beispiel:

Infolge von +U oder U wird sämtlicher Text im Puffer in Großschrift umgewandelt. -U deaktiviert diesen Vorgang wieder.

-V schaltet die Anzeige der Zeilennummern ab und +V wieder ein. 0V zeigt die Größe des noch freien und des gesamten Pufferspeichers in Bytes an.

Beim CPC erscheint infolge von OV:

2 4 8 6 3 / 2 5 0 1 5

Dies besagt, daß der Texpuffer insgesamt 25015 Bytes umfaßt, von denen 24863 Bytes noch nicht belegt sind.

nW

Vom Textzeiger an werden n Zeilen in die Zwischendatei mit dem Namen der zu bearbeitenden Datei und der Extension $$$ geschrieben. Ohne Zahlenangabe wird n=1 angenommen. Die Zwischendatei kann mit dem R-Befehl wieder gelesen werden.

**nX**

Vom Textzeiger an werden n Zeilen in die Zwischendatei X$$$$$$$.LIB geschrieben, die mit dem R-Befehl wieder gelesen werden kann. Ist n=0, wird die Zwischendatei gelöscht.

***nZ***

Vor der Ausführung eines ED-Befehls wird eine Pause von n Zeiteinheiten eingelegt.

**ED**-Steuerzeichen

| | |
|---|---|
| CTRL¯C | Warmstart und Rückkehr zu CP/M |



| | |
|---|---|
| CTRL¯E | Fortsetzung der Befehlszeile am Anfang der nächsten Bildschirmzeile |
| CTRL¯H | Löschen des Zeichens links vom Cursor, dabei Cursor um eine Stelle nach links ( Backspace) |
| CTRL¯I | Cursor um eine Tabulatorbreite (7 Zeichen) vorrücken |
| CTRL–J | Entspricht Drücken der ENTER-Taste |
| CTRL–L | Dient zum Austauschen von RETURN-Zeichen bei Suchvorgängen |
| CTRL¯M | Entspricht Drücken der ENTER-Taste (Carriage Return) |
| CTRL¯R | Erneute Anzeige der Befehlszeile nach Fehlerkorrektur |
| CTRL–U | Befehlszeile löschen, Cursor in nächste Zeile |
| CTRL–X | Befehlszeile löschen, Cursor zurück an Zeilenanfang |
| CTRL–Z | Dient für Such- und Austauschoperationen sowie zur Beendigung des Einfügemodus (I) |

## ERA

Residenter Standard-CP/M 2.2-Befehl

Beschreibung:

ERA dient zum Löschen einzelner Dateien oder Dateigruppen. Davon ausgenommen sind schreibgeschützte (R/O) Dateien. Wird der Versuch unternommen, eine schreibgeschützte Datei zu löschen, erscheint ein Bdos-Error, der nur durch einen Warmstart durch <CTRL—C> zu beheben ist.

Aufruf:

```
ERA (Laufwerk:) Dateiname
```

Dabei kann der Dateiname ein- oder mehrdeutig sein.

Beispiele:

```
ERA PROBE.TXT
```

löscht die Datei PROBE.TXT im Bezugslaufwerk und

```
ERA B:PROBE.TXT
```

löscht die gleiche Datei, wenn sie in Laufwerk B abgelegt ist.

```
ERA *.DAT
```

löscht alle Dateien mit der Extension DAT und

```
ERA *.*
```

löscht sämtliche Dateien auf der Diskette. Da dieser Befehl bei falscher Anwendung fatale Folgen haben kann, er-scheint vor der Ausführung noch die Sicherheitsabfrage

```
ALL (Y/N)?
```

Wird hier Y für "Ja" eingegeben, wird die gesamte Diskette gelöscht, ansonsten erscheint wieder das Anforderungs-zeichen.

Wird die angegebene Datei oder Dateigruppe nicht gefunden, erscheint die Meldung:

```
NO FILE
```



## FILECOPY

Systemspezifischer CPC-Befehl (Datei FILECOPY.COM)

Beschreibung:

FILECOPY dient zum Kopieren beliebiger Dateien oder Dateigruppen auf eine andere Diskette, wo sie unter gleichem Namen abgelegt werden. FILECOPY arbeitet nur mit Laufwerk A und liest die Dateien in den Arbeitsspeicher. Nach Austausch der Disketten werden sie dann auf die Zieldiskette geschrieben.

Bei mehrdeutigen (ambigiuous) Dateinamen erfolgt zu-nächst eine Abfrage, ob alle Dateien der angegebenen Dateigruppe kopiert werden sollen. Wenn ja, wird jede Datei einzeln kopiert, wozu jeweils ein Diskettenwechsel vorzunehmen ist. Wenn nein, werden sämtliche Dateien einzeln auf dem Bildschirm angezeigt und jeweils abgefragt, ob sie kopiert werden sollen.

Aufruf:

```
FILECOPY Dateiname
```

Beispiele:

```
FILECOPY STAT.COM
```

kopiert die Datei STAT.COM,

```
FILECOPY *.TXT
```

kopiert alle Dateien mit der Extension TXT und

```
FILECOPY *.*
```

sämtliche Dateien, die sich auf der Diskette befinden.

In den letzten beiden Beispielen findet vorher noch die Abfrage statt, ob alle Dateien dieser Dateigruppe kopiert werden sollen.

# FORMAT

Systemspezifischer CPC-Befehl (Datei FORMAT.COM)

Beschreibung:

FORMAT dient zum Formatieren von Disketten und arbeitet nur mit Laufwerk A. Dabei besteht die Möglichkeit, Disketten wahlweise im Systemformat, Datenformat, Vendorformat oder IBM-Format zu formatieren.

Bei der Formatierung im Systemformat wird das CP/M-Betriebsystem mit in die äußeren beiden Spuren kopiert. Beim Vendorformat werden diese beiden Spuren zwar reserviert, aber das Betriebssystem nicht hineingeschrieben.

Beim Datenformat dagegen findet keine Reservierung der beiden äußeren Spuren statt, so daß sie zur Ablage von Dateien zur Verfügung stehen. Dadurch nimmt die freie Kapazität der Diskette etwas zu.

Das IBM-Format dient in erster Linie zum Datenaustausch mit anderen Computern, die dieses Format ebenfalls lesen bzw. beschreiben können und sollte nur zu diesem Zweck verwendet werden.

Aufruf:

`FORMAT`
formatiert die Diskette im Systemformat,

`FORMAT V`

formatiert die Diskette im Vendorformat,

`FORMAT D`

formatiert die Diskette im Datenformat und

`FORMAT I`

formatiert die Diskette im IBM-Format.

Nach dem jeweiligen Aufruf erscheint die Aufforderung, die zu formatierende Diskette in Laufwerk A einzulegen.



# LOAD

Nichtresidenter Standard-CP/M 2.2-Befehl (Datei LOAD.COM)

Beschreibung:

LOAD liest eine HEX-Datei im Intel-Hex-Format, die durch den Assembler ASM erzeugt wurde und formt sie in eine ausführbare COM-Datei um, die auf Diskette geschrieben wird.

Aufruf:

`LOAD (Laufwerk:) Dateiname HEX¯Datei`

Der Dateiname muß ohne Extension eingeben werden.

Beispiel:

`LOAD MUSTER`

liest die Intel-Hex-Datei MUSTER.HEX und wandelt sie in die Datei MUSTER.COM um, die von CP/M aus aus-führbar ist.

# MOVCPM

Nichtresidenter Standard-CP/M 2.2-Befehl (Datei MOVCPM.COM)

Beschreibung:

MOVCPM paßt das CP/M-Betriebssystem an die Größe des verfügbaren RAM-Speichers an (beim CPC normaler-weise nicht erforderlich).

Aufruf:

```
MOVCPM
```

verschiebt das Betriebssystem in den obersten Bereich des RAM-Speichers und startet es sogleich (funktioniert beim CPC nicht).

```
MOVCPM n
```

erzeugt ein Betriebssystem für einen Speicher von der Größe n mal 256 Bytes und startet es sogleich.

Beispiel:

```
MOVCPM 96
```

erzeugt ein Betriebssystem von 96*256 Bytes = 24 KBytes und startet es.

Soll das verschobene Betriebssystem nicht gestartet, sondern mit SYSGEN oder SAVE erst abgespeichert werden, ist

```
MOVCPM * *
```

für den größtmöglichen Speicher (funktioniert beim CPC nicht) oder

```
MOOVCPM n *
```

für einen Speicher von der Größe n mal 256 Bytes einzugeben. Es kann anschließend mit

```
SYSGEN *
```

auf die Systemdiskette übertragen werden.



# PIP

Nichtresidenter Standard-CP/M 2.2-Befehl (Datei PIP.COM)

Beschreibung:

PIP ist ein universelles Datei-Übertragungsprogramm von Diskette zu Diskette einerseits und von Diskette zu Peripheriegeräten (oder umgekehrt) über die CP/M-Ein-/Ausgabekanäle andererseits.

Grundsätzlich können mit PIP alle Arten von Dateien übertragen werden, ohne Rücksicht auf ihren Inhalt oder ihre Beschaffenheit. Jedoch sind viele Befehle nur für ASCII-Text-Dateien, die mit einem `CTRL—Z (1AH)` ab-schließen, geeignet.

Anwendungs-beispiele:

Kopieren einzelner Dateien
Kopieren ganzer Disketteninhalte
Dateien aneinanderreihen
Listen von ASCII-Dateien auf Bildschirm oder Drucker
Verändern von Dateien während des Kopiervorgangs
Auszüge von Textdateien herstellen

Aufruf:

```
PIP   (Befehlsfolge)
```

oder

```
PIP
* (Befehlsfolge 1)
* (Befehlsfolge 2)
* (Befehlsfolge 3)
```

```
* (Befehlsfolge n)
* <ENTER> (PIP verlassen mit Warmstart)
```

Jede Befehlsfolge ist in sich abgeschlossen und wird einzeln ausgeführt.
Allgemeine Form der Befehlsfolge:

```
Ziel = Quelle [Option]
```

Dabei ist Ziel

(Laufwerk:) Dateiname

oder ein Ausgabekanal auf Peripheriegerät:

PUN : oder TTY:

und Quelle ist

(Laufwerk:) Dateiname

oder

(Laufwerk:) Dateinamel, (Laufwerk:) Dateinamel, ...
... (Laufwerk:) Dateiname n

oder ein Eingabekanal von Peripheriegerät:

RDR: oder LST:

**Standardbelegung der Ein-/Ausgabekanäle:**

| | | |
|---|---|---|
| CON: (Konsole) | ==> | CRT: (Bildschirm u. Tastatur) |
| RDR: (Lochstreifenleser) | ==> | TTY: (Fernschreiber) |
| PUN: (Lochstreifenstanzer) | ==> | TTY: (Fernschreiber) |
| LST: (Listeinheit) | | LPT: (Centronics-Schnittstelle bzw. Drucker) |

Dem RDR:- und PUN:-Kanal ist zwar TTY: zugeordnet, aber in Wirklichkeit ist beim CPC kein Gerät angeschlossen. Deshalb steht TTY: für Erweiterungen (z.B. serielle Schnittstelle) zur Verfügung.

Beispiele **für Befehlsfolge (ohne Optionen):**

DATEINEU. TXT=DATEIALT. TXT

kopiert die Datei DATEIALT.TXT, die im Bezugslaufwerk abgelegt ist, in die Datei DATEINEU:TXT, die ebenfalls im Bezugslaufwerk angelegt wird.

B:DATEI1.COM=A:DATEI1.BAK



kopiert die Datei DATEII.BAK von Laufwerk A nach Laufwerk B, wo sie unter dem Namen DATEI1.COM abgelegt wird.

.A:=B: LAGER.DAT

kopiert die Datei LAGER.DAT von Laufwerk B nach Laufwerk A, wo sie unter demselben Namen abgelegt wird.

B:=A:*.COM

kopiert sämtliche COM-Dateien von Laufwerk A nach Laufwerk B.

B:=A*.*

kopiert sämtliche Dateien von Laufwerk A nach Laufwerk B.

**B:DATEI.TXT=A:DATEI1.TXT,A:DATEI2.TXT,** B:
DATEI3.TXT

bildet die Datei DATEI.TXT in Laufwerk B durch Aneinanderreihen der Dateien DATEI1.TXT und DATEI2.TXT in Laufwerk A sowie der Datei DATEI3.TXT in Laufwerk B.

LST:=PROBE.TXT

gibt die Datei PROBE.TXT im Bezugslaufwerk auf dem Drucker aus.

CON:=B:TEST1.DAT,B:TEST2.DAT

listet die Dateien TESTI.DAT und TEST2.DAT, die sich beide in Laufwerk B befinden, auf dem Bildschirm.

## Optionen:

[B1

Dateien werden blockweise übertragen, wobei die Größe der Blöcke von der jeweiligen Größe des Arbeitsspeichers

abhängt. Diese Maßnahme ist dann sinnvoll, wenn z.B. über ein Modem oder einen Akustikkoppler mit serieller RS232-Schnittstelle Dateien ein- oder ausgegeben werden, da die Schnittstelle nicht gleichzeitig mit den Floppylaufwerken betrieben werden kann.

Bei der übertragung werden die Daten zunächst in einen Puffer im Arbeitsspeicher geschrieben, bis ein XOFF-Zeichen (CTRL¯S) erscheint und erst dann in die Zieldatei oder auf das Zielgerät übertragen. Der gleiche Vorgang wiederholt sich mit dem nächsten Block und zwar solange, bis ein CTRL–Z-(Textende-)Zeichen erscheint.

Bei der Datenfernübertragung spricht man in diesem Zusammenhang auch von Upload und Download ( Zwischenspeicherung von Daten vor Ausgabe auf bzw. nach Eingabe von der Schnittstelle).

Beispiel:

DATEN. TXT=RDR: [B]

Daten werden über die Schnittstelle, die hier dem RDR:-Kanal zugeordnet ist, blockweise eingelesen und in die Datei DATEN.TXT geschrieben.

**[Dn]**

Beispiel:

Bei Angabe dieser Option werden Textdateien nur bis zu dem Zeichen n pro Zeile übertragen. Dies hat beispielsweise den Vorteil, daß nur soviele Zeichen in eine Bild-schirm- oder Druckerzeile übertragen werden, wie darin Platz ist. Die Zeilen werden dabei hinter dem n-ten Zeichen abgeschnitten.

LST:=LISTE.TXT[D35]

Von der Datei LISTE.TXT weden jeweils nur 35 Zeichen pro Zeile auf dem Drucker ausgegeben, wobei die restlichen Zeichen unterdrückt werden.



[E1

Dies ist eine Echofunktion, die Textdateien während ihrer Übertragung auf dem Bildschirm auflistet. Man kann somit überprüfen, ob die Datei richtig übertragen wird.

Beispiel:

B:DATEN2.TXT=A:DATEN1.TXT[E]

Die Datei DATEN1.TXT in Laufwerk A wird in die Datei DATEN2.TXT in Laufwerk B kopiert und gleichzeitig auf dem Bildschirm ausgegeben.

[FI

Diese Option entfernt bei der Ubertragung sämtliche Formfeed-Zeichen (OCH bzw. CTRL–L) in Textdateien, die auf dem Drucker einen Seitenvorschub veranlassen.

Beispiel:

CON:=PROBE .TXT[F]

In der Datei PROBE.TXT werden sämtliche Formfeed-Zeichen entfernt, während sie auf dem Bildschirm ausgegeben wird. Für die Bildschirmausgabe sind diese Zeichen nicht sinnvoll, da sie jedesmal den Bildschirm löschen.

[GnI

Kopiert Dateien aus dem angegebenen Benutzerbereich n in den gegenwärtigen Benutzerbereich.

Beispiel:

TEXT .DAT=BRIEF.BAK[G3]

Die Datei BRIEF.BAK, die im Benutzerbereich 3 abgelegt ist, wird in den aktuellen Bereich kopiert. Falls nichts anderes angegeben, handelt es sich dabei immer um den Standard-Bereich 0.

[HI

Prüft nach, ob sich die übertragenden Daten im Intel-Hex-Format befinden. Diese Option spielt nur bei der Assemblerprogrammierung eine Rolle.

**[11**
[L1

Übergeht bei der übertragung von Intel-Hex-Dateien sämtliche Nullbytes. Diese Option spielt nur bei der Assemblerprogrammierung eine Rolle.

Wandelt Textdateien während der Übertragung in Kleinbuchstaben um.

Beispiel:

```
CON:=PROBE .TXT[L]
```

Die Datei PROBE.TXT wird auf dem Bildschirm ausgegeben und erscheint dabei nur in Kleinschrift.

[N1

Jeweils am Zeilenanfang von Textdateien werden Zeilennummern gesetzt.

Beispiel:

```
CON:=BRIEF .TXT[N]
```

Die Datei BRIEF.TXT wird auf dem Bildschirm ausgegeben und erscheint dabei in folgender Form:

```
1:      (1. Zeile)
2:      (2. Zeile)
3:      (3.
Zeile) usw.
```

[N21

Jeweils am Zeilenanfang von Textdateien werden sechsstellige Zeilennummern mit führenden Nullen gesetzt.



Beispiel:

```
CON:=BRIEF.TXT[N2]
```

Die Datei BRIEF.TXT wird auf dem Bildschirm ausgegeben und erscheint dabei in folgender Form:

```
000001
(1. Zeile)
000002
(2. Zeile)
000003
```

**[O1**

Diese Option ist nur dann notwendig, wenn andere Dateien als Textdateien miteienander verbunden werden sollen, wobei keine Abfrage auf das Textendezeichen (CTRL—Z) stattfindet.

Beispiel:

```
DATE INEU.BIN=DATEI 1. BIN, DATEI 2.
BIN,DATEI 3 .BIN[O]
```

Die Dateien DATEI1.BIN, DATEI2.BIN und DATEI3.BIN werden hintereinander in die Datei DATEINEU.BIN geschrieben, wobei jeweils die volle physikalische Länge ohne Abfrage auf CTRL—Z übertragen wird.

**[Pn]**

Diese Option fügt alle n Zeilen ein Formfeed-Zeichen (OCH bzw. CTRL—L) in eine Texdatei ein. Formfeed-Zeichen sorgen bei der Druckerausgabe für einen Seitenvorschub.

Beispiel:

```
LST:=PROBE.TXT[P64]
```

Bei der Ausgabe der Datei PROBE.TXT über den Drucker findet alle 64 Zeilen ein Seitenvorschub statt.

**[Q"Begriff"^Z1**

Mit dieser Option wird eine Textdatei bis zu der Stelle übertragen, an der ein vorgegebener Suchbegriff erscheint.

Dabei wird der "Begriff" selbst mitübertragen. Die Option sollte nur nach separatem Starten von PIP erteilt werden, wenn das Sternchen zur Befehlseingabe auffordert.

Beispiel:

```
CON: =LAGER. DAT[QStiefel^Z]
```

Die Datei LAGER.DAT wird auf dem Bildschirm aus-gegeben, und zwar bis zu der Stelle, an der der Begriff " Stiefel" auftritt.

[Rl

Dient zum Kopieren von Systemdateien.

Beispiel:

```
B:=A:GEHEIM.DAT[R]
```

[S"Begriff"^Z]

Beispiel:

Die Systemdatei GEHEIM.DAT wird von Laufwerk A nach Laufwerk B übertragen, wo sie ebenfalls als Systemdatei abgelegt wird.

Mit dieser Option wird eine Textdatei ab der Stelle übertragen, an der ein vorgegebener Suchbegriff erscheint. Dabei wird der "Begriff" selbst mitübertragen. Die Option sollte nur nach separatem Starten von PIP erteilt werden, wenn das Sternchen zur Befehlseingabe auffordert.

```
CON:=LAGER.DAT[SHemd^Z]
```

Die Datei LAGER.DAT wird auf dem Bildschirm aus-gegeben und zwar ab der Stelle, an der der Begriff "Hemd" auftritt.

**[Tu]**

Setzt Tabulator für Ausdrucke, wobei n der Spaltenbreite in Zeichen entspricht.



[UI

Wandelt Textdateien während der übertragung in Grußbuchstaben um.

Beispiel:

```
CON:=PROBE .TXT[U]
```

Die Datei PROBE.TXT wird auf dem Bildschirm aus-gegeben und erscheint dabei nur in Großschrift.

Während der Übertragung von Diskettendateien führt diese Option ein Verify durch. Nach der normalen Übertragung wird dabei die Ziel- mit der Quelldatei nochmals verglichen. Bei fehlerhafter bertragung erscheint die Meldung VERIFY ERROR (Befehlsfolge).

Beispiel:

```
B:=A:*.TXT[V]
```

Hier werden sämtliche TXT-Dateien von Laufwerk A nach Laufwerk B mit Verify übertragen.

[WI

Mit dieser Option werden schreibgeschützte Dateien (R/O) ohne Sicherheitsfrage überschrieben.

Beispiel:

```
B:=A:PROTECT.COM[W]
```

Die Datei PROTECT.COM wird von Laufwerk A nach Laufwerk B übertragen und überschreibt dort eine möglicherweise vorhandene Datei unter gleichem Namen, selbst wenn diese schreibgeschützt ist.

[Z[

Die Option löscht bei der Übertragung von ASCII-Dateien das höchstwertige Bit (Bit *7)* und wird nur für spezielle Anwendungsfälle eingesetzt.

**Besondere Gerätezuordnung:**

Neben den standardmäßigen Ein-/Ausgabekanälen, CON:, RDR:, PUN: und LST: (s.o.) können bei PIP noch folgende Gerätebezeichungen verwendet werden:

**PRN:**

PRN: verwendet den LST:-Kanal, über den in der Regel der Drucker angeschlossen ist, zum Ausdrucken von Texten. Dabei werden jedoch folgende Optionen automatisch gesetzt, sofern nichts anderes vorgegeben ist:

Seitenvorschub alle 60 Zeilen
Tabulator auf 8 Positionen
Numerierung der Textzeilen

Diese Eigenschaften entsprechen den Optionen [NPT8] (s.o.).

Beispiel:

```
PRN:=DATEN.TXT
```

druckt die Datei DATEN.TXT mit den oben angegebenen Optionen aus.

**INP:**

Benutzerdefiniertes Eingabegerät, das über einen Vektor in der Speicherstelle 103H angesprochen werden kann. Einzulesende Zeichen sind dabei an der Adresse 109H abzulegen. Um diese Geräteeinheit zu benutzen, muß sich der Anwender jedoch eine eigene Treiberroutine schreiben.

**OUT:**

Benutzerdefiniertes Ausgabegerät, das über einen Vektor in der Speicherstelle 106H angesprochen werden kann. Auszugebende Zeichen sind dabei im Register C der CPU abzulegen. Um diese Geräteeinheit zu benutzen, muß sich der Anwender jedoch eine eigene Treiberroutine schreiben.



## REN

Residenter Standard-CP/M 2.2-Befehl

Beschreibung:

REN dient zum Umbenennen einzelner Dateien. Mehrdeutige Dateinamen sind nicht zugelassen, auch darf noch keine andere Datei unter dem neuen Namen auf der Diskette abgelegt sein.

Aufruf:

```
REN (Laufwerk:) Datei neu = Datei alt
```

Beispiel:

```
REN ARTIKEL.TXT=LAGER.DAT
```

nennt im Bezugslaufwerk die Datei LAGER.DAT in ARTIKEL.TXT um. Das gleiche geschieht mit

```
REN B: ARTIKEL.TXT=LAGER.DAT
```

## SAVE

Residenter Standard CP/M 2.2-Befehl.

Beschreibung:

Speichert einen Speicherbereich auf Diskette ab. Der Bereich beginnt immer bei 100H und wird in 256 Byte-Einheiten angegeben. Eine Laufwerksangabe ist hier nicht möglich, denn SAVE arbeitet nur mit dem Bezugslaufwerk.

Der häufigste Anwendungsfall von SAVE ist das Wiederabspeichern einer Datei, die mit DDT (s.o.) geändert wurde. Dabei wird die Datei zunächst mit DDT in den Speicher ab 100H geladen. Nach der Berabeitung wird mit CTRL—C oder dem DDT-Befehl GO ein Warmstart aus-gelöst, so daß das Anforderungszeichen wieder erscheint. Jetzt kann die Datei mit SAVE auf Diskette geschrieben werden.

Aufruf:

**SAVE** `n Dateiname`

wobei n den Speicherbereich in 256 Byte-Einheiten angibt.

Beispiel:

Eine Datei wurde mit DDT geändert und soll unter dem Namen RECHNUNG.COM wieder auf Diskette im Bezugslaufwerk geschrieben werden. Als sie vorher mit DDT eingelesen wurde, erschien die Meldung:

```
NEXT PC
5B80 0100
```

Diese Meldung sollte unbedingt notiert werden, da sie für SAVE benötigt wird. Die Datei nimmt demnach den Speicherbereich zwischen O100H und 5B80H ein und belegt somit

`5B80H — O100H = 5A80H Bytes`

Dieser hexadezimale Wert muß zunächst in einen dezimalen umgerechnet werden, wobei die ersten beiden Stellen die Anzahl der 256 Byte-Einheiten angeben. Folglich ist

`5AH = 5*16 + 10 =` 90 Einheiten zu 256 Bytes.



Da die letzten beiden Stellen ungleich Null sind, ist noch eine Einheit dazuzuzählen, so daß insgesamt 91 Einheiten abzuspeichern sind.

Jetzt kann die Datei mit `SAVE 91`

`RECHNUNG.COM` auf Diskette

geschrieben werden.

# SETUP

Gerätespezifischer CPC-Befehl (Datei SETUP.COM)

Beschreibung:

SETUP bietet die Möglichkeit, verschiedene Grundeinstellungen am CPC für den CP/M-Betrieb vorzunehmen. Die eingegebenen Werte werden dann im BOOT-Sektor gespeichert und dienen beim CP/M-Aufruf (Kaltstart) zur Konfiguration des Systems.

SETUP ist menügesteuert und fragt, ob die vorbelegten Werte beibehalten werden sollen. Wenn ja, erscheint der nächste Menüpunkt, ansonsten erfolgt zunächst die Aufforderung, entsprechende Werte einzugeben.

Nach Abfrage sämtlicher Menüpunkte kann der Bediener entscheiden, ob die geänderten Werte im BOOT-Sektor abgelegt werden sollen oder nicht.

SETUP sollte zunächst auf die Diskette kopiert werden, auf der der BOOT-Sektor geändert werden soll und ist dann im Bezugslaufwerk mit
SETUP

aufzurufen. Daraufhin beginnt die Abfrage der einzelnen Werte. Nachfolgend die wichtigsten Parameter, die geändert werden können:

**Initial Command Buffer**

Bietet die Möglichkeit, direkt nach dem Aufruf von CP/M einen CP/M-Befehl, eine COM-Datei oder eine SUBMITDatei auszuführen. Jede dieser Anweisungen ist mit ^AM abzuschließen, was dem Drücken der ENTER-Taste entspricht.

Beispiele:

DIR^AM
listet nach dem CP/M-Aufruf das Directory, WS

^AM

führt automatisch das Programm WS.COM aus oder



SUBMIT LIST^AM

führt die Anweisungen in der Befehlsdatei LIST.SUB aus.

**Sign-on-string**

Dient zur Modifizierung der CP/M-Einschaltmeldung, wobei ein beliebiger String vorgegeben werden kann. Dieser muß nicht unbedingt ausschließlich aus Textzeichen bestehen, sondern kann auch beliebige Bildschirmsteuerzeichen (siehe Bedienerhandbuch) enthalten, die zum Beispiel die Bildschirmfarben setzen. Solche Zeichen sind aber zu umschreiben, damit sie in einem String dargestellt werden können, beispielsweise durch CTRL-Steuerzeichen oder durch Textzeichen mit äquivalentem ASCII-Code. Ein Beispiel hierfür bietet der String der Standard-Einschaltmeldung.

Beispiel:

Guten Tag, CP/M meldet sich zur Stelle^J^M

gibt eine Einschaltmeldung mit diesem Text aus, wobei der String mit ^AJ^M (CR, LF) abgeschlossen sein muß.

**Keyboard tr'anslation set**

Hiermit kann die Tastatur neu belegt werden, ähnlich wie mit dem BASIC-KEYDEF-Befehl, was durch die entsprechende Zuordnung der Tastennummern zu den ASCII-Codes geschieht.

**Keyboard expansions set**

Dient zur Belegung der Funktionstasten, ähnlich wie beim BASIC-KEY-Befehl, wobei die Zuordnung der Tasten zu dem entsprechenden Befehlswort oder Steuerzeichen anzugeben ist.

Default IO byte settings

Dient zur Änderung der Standard-Gerätezuordnung für die Ein-/Ausgabekanäle, wobei die gleichen Bezeichnungen wie bei STAT (s.u.) gelten.

# STAT

Nichtresidenter Standard-CP/M 2.2-Befehl (Datei STAT.COM)

Beschreibung:

STAT ist ein vielseitiges CP/M-Dienstprogramm, mit dessen Hilfe man Informationen über Dateien und Disketten abfragen und verschiedene Parameter ändern kann. Im einzelnen handelt es sich dabei um:

Ermittlung des augenblicklichen Diskettenzustandes
Ermittlung des Zustandes von Dateien
Setzen und Aufheben eines Schreibschutzes für Dateien
Dateien zu Systemdateien erklären und umgekehrt
Ermittlung der Diskettenparameter
Laufwerk mit Schreibschutz versehen
Ermittlung und Anderung der Gerätezuordnungen zu den Ein- und Ausgabekanälen
Ermittlung der augenblicklichen Benutzernummer

Aufruf:

```
STAT (Laufwerk:) Argument
```

Das Argument ergibt sich dabei aus den folgenden Erläuterungen.

**Ermittlung des augenblicklichen Diskettenzustandes:**

Zu diesem Zweck ist lediglich `STAT`

(ohne Argument) einzugeben, wodurch sämtliche angeschlossenen Laufwerke abgefragt werden. Dabei er-scheint z.B.:

```
A: R/W, Space 56k
B: R/0, Space 41k
```

Dies besagt, daß Laufwerk A nicht schreibgeschützt (R/W) ist und noch über 56K freie Kapazität zur Ablage von Dateien verfügt. Laufwerk B dagegen ist schreibgeschützt (R/O) und hat noch 41K zur Verfügung.

Bei Angabe eines Laufwerkes erscheint nur die freie Diskettenkapazität.



Beispiel:

```
STAT B:
Bytes Remaining On B: 41k
```

Ermittlung des Zustandes **von Dateien**

Zur Ermittlung des Dateizustandes ist ein ein- oder mehrdeutiger Dateiname in der Form von

```
STAT (Laufwerk:) Dateiname
```

mit anzugeben.

Beispiele:

```
STAT PIP.COM
```

Darauf erscheint:

```
Recs Bytes Ext Acc
58     8k     1 R/W A:PIP.COM
Bytes Remaining On A: 44k
```

wobei

| | |
|---|---|
| Recs | Anzahl der belegten Records (128 Bytes) |
| Bytes | Umfang der Datei in Bytes bzw. Kilobytes |
| Ext | Anzahl der belegten Extents (16k) |
| Acc | Datei ohne (R/W) bzw. mit Schreibschutz (R/O) |

Schreibschutz **für Dateien setzen/aufheben**

Dateien werden mit

```
STAT (Laufwerk:) Dateiname $R/O
```

schreibgeschützt und stehen daraufhin nur noch für Lesezugriffe zur Verfügung. Im Directory sind sie dann mit dem R/O-Attribut gekennzeichnet. Umgekehrt hebt

```
STAT (Laufwerk:) Dateiname $R/W
```

den Schreibschutz wieder auf.

Beide Anweisungen beziehen sich nicht nur auf eindeutige, sondern ebenfalls auf mehrdeutige Dateinamen, so daß auch ganze Dateigruppen geschützt werden können.

Beispiele:

```
STAT DATEN.TXT $R/O
```

versieht die Datei DATEN.TXT, die im Bezugslaufwerk abgelegt ist, mit einem Schreibschutz.

```
STAT B:PROBE.SUB $R/W
```

hebt den Schreibschutz für die Datei PROBE.SUB in Laufwerk B wieder auf.

```
STAT *.* $R/O
```

versieht sämtliche Dateien im Bezugslaufwerk mit einem Schreibschutz.

**Dateien *zu*** Systemdateien **erklären und umgekehrt**

Unter Systemdateien versteht man Dateien, die mit dem DIR-Befehl nicht auflistbar und normalerweise auch nicht kopierbar sind. Der Befehl

```
STAT (Laufwerk:) Dateiname $SYS
```

erklärt eine Datei zur Systemdatei und

```
STAT (Laufwerk:) Dateiname $DIR
```

macht diesen Vorgang wieder rückgängig.

Da Systemdateien nicht mit DIR aufgelistet werden können, ist dies nur mit

```
STAT (Laufwerk:) Dateiname
```

`(s.o.)` möglich. Der Dateiname erscheint in Klammern.

Beispiele:

```
STAT B:GEHEIM.DAT $SYS
```

erklärt die Datei GEHEIM.DAT, die in Laufwerk B angelegt ist, zur Systemdatei, so daß sie nicht mehr im Directory erscheint. Mit

```
STAT B:GEHEIM.DAT $DIR
```

wird dieser Vorgang wieder rückgängig gemacht und die Datei ist mit dem DIR-Befehl auflistbar.



**Ermittlung der Diskettenparameter**

Der Befehl

```
STAT DSK:
```

gibt für jedes angeschlossene Laufwerk die Speicheraufteilung auf der Diskette an. Beim CPC erhält man zumeist folgende Angaben:

```
   A: Drive Characteristics
1368: 128 Byte Record Capicity
 171: Kilobyte Drive Capicity
  64: 32 Byte Directory Entries
  64: Checked Directory Entries
 128: Records/ Extent
   8: Records/ Block
  36: Sektors/ Track
   2: Reserved Tracks
```

Diese Parameter haben folgende Bedeutung (in der Reihenfolge ihrer Anzeige):

- Diskettenparameter für Laufwerk A:
- Gesamtkapazität 1368 Records zu je 128 Bytes für Dateien (ohne Systemspuren)
- 171K Gesamtkapazität zur Ablage von Dateien
- Platz für 64 Directory-Einträge zu je 32 Bytes
- Sämtliche 64 Directory-Einträge können mit einem Schreibschutz versehen sehen
- In einem Extent (Directory-Eintrag) können 128 Records (16K) verwaltet werden
- In einem Block (1024 Bytes) befinden sich 8 Records
- Auf einer Diskettenspur sind 36 Records (nicht Sektoren!) abgelegt
- 2 reservierte Spuren für die Ablage des CP/M-Betriebssystems und des BOOT-Sektors

Laufwerk mit **Schreibschutz versehen**

Unabhängig von einem Schreibschutz für Dateien kann auch ein Laufwerk vor Schreibzugriffen geschützt werden. Dabei handelt es sich allerdings nur um einen zeitweiligen Schutz, der nicht auf der Diskette aufgetragen wird.

```
STAT (Laufwerk:) = R/O
```

stellt die Anweisung in allgemeiner Form dar.

Beispiel:

```
STAT B:=R/O
```

setzt einen Schreibschutz für Laufwerk B.

Der Schreibschutz für Laufwerke kann nur mit einem Warmstart (CTRL¯C) wieder aufgehoben werden.

**Gerätezuordnungen**

CP/M 2.2 unterhält insgesamt vier Ein- und Ausgabekanäle, denen jeweils ein externes Gerät zugeordnet ist. Dies gilt auch für die Tastatur und den Bildschirm, die unter CP/M als Konsole bezeichnet werden.

Die jeweiligen Gerätezuordnungen sind folgendermaßen abzufragen:

```
STAT DEV:
```

Beim CPC erscheinen daraufhin diese Standard-Zuordnungen:

| | |
|---|---|
| CON:=CTR: | (Konsolenkanal) |
| RDR:=TTY: | (Kanal für Lochstreifenleser) |
| PUN:=TTY: | (Kanal für Lochstreifenstanzer) |
| LST: =LPT : | (Kanal für List-Gerät) |

Die Zuordnung kann geändert werden durch:

```
STAT Kanal: = Gerät:
```

Beispiel:

```
STAT LST:=TTY:
```

ordnet dem List-Kanal einen Fernschreiberausgang oder eine serielle Schnittstelle zu.

Folgende Geräte können den einzelnen Kanälen zugeordnet



TTY: Fernschreiber
BAT: Stapelverarbeitung (beim CPC ohne Bedeutung)
UC1: Vom Benutzer selbst definiertes Terminal
RDR: (Kanal für Lochstreifenleser)

TTY: Fernschreiber
PTR: Besonders schneller Lochstreifenleser
UR1: Vom Benutzer definiertes Lesegerät
UR2: Vom Benutzer definiertes Lesegerät
**PUN:** (Kanal für Lochstreifenstanzer)

TTY: Fernschreiber
PTB: Besonders schnelles Stanzgerät
UP1: Vom Benutzer definiertes Ausgabegerät
UP2: Vom Benutzer definiertes Ausgabegerät
LST: (Listkanal)

LPT: Drucker
TTY: Fernschreiber
CRT: Bildschirm
UL1: Vom Benutzer definiertes Ausgabegerät

Benutzerbereiche ermitteln

```
STAT USR:
```

ermittelt die gegenwärtige Benutzernummer sowie die Nummern aller Benutzer, die Dateien im Bezugslaufwerk abgelegt haben.

Beispiel:

```
Active User: 5
Active Files: 2 3 5 7 12
```

Diese Angabe besagt, daß der Benutzerbereich 5 aktiv ist. Darüber hinaus sind Dateien in den Bereichen 2, 3, 5, 7 und 12 abgelegt.

# SUBMIT

Nichtresidenter Standard-CP/M 2.2-Befehl (Datei SUBMIT.COM)

Beschreibung:

SUBMIT arbeitet eine Befehlsdatei (Stapeldatei) ab, die mehrere CP/M-Befehls- oder Programmaufrufe enthalten kann. Die Befehlsdatei muß die Extension SUB tragen und ist eine reine ASCII-Datei, die mit jedem Texteditor erstellt werden kann.

Vor der Ausführung der einzelnen Befehle legt SUBMIT zunächst die Zwischendatei $$$.SUB an, in die die Befehlsdatei kopiert wird. Bei der Abarbeitung wird jeder Befehl dann aus der Zwischendatei herausgenommen, bis die Datei leer ist. Dann sind sämtliche Befehle abgearbeitet und die Zwischendatei wird wieder gelöscht.

Die einzelnen Befehle erscheinen während ihrer Ausführung auf dem Bildschirm, und zwar genauso, als wären sie von Hand eingegeben.

Aufruf:

```
StTBMIT Dateiname—SUB—Datei Parameter
```

Der Dateiname der SUB-Datei muß ohne die Extension SUB angegeben werden.

Beispiel:

Mit Hilfe von SUBMIT sollen die CP/M-Befehle

```
DIR
REN B: LAGER .DAT=LAGER.BAK
BESTAND     (Ausführung der Datei BESTAND.COM)
STAT B: LAGER.*
```

ausgeführt werden. Dazu sind sie zeilenweise mit einem Texteditor (z.B. ED) in eine Datei mit der Extension SUB (z.B. INVENTUR.SUB) zu schreiben. Infolge von

```
SUBMIT INVENTUR
```

werden schließlich die einzelnen Befehle der Reihe nach ausgeführt.

Nachfolgend das gleiche Beispiel, jedoch unter Verwendung von Variablen. Für LAGER wird die Variable $1



und für BESTAND die Variable $2 eingesetzt. Die Datei INVENTUR.SUB sähe dann so aus:

```
DIR
REN B:$1.DAT=$1.BAK
$2
STAT B:$1.*
```

Die Befehlsdatei ist dann mit

```
SUBMIT INVENTUR LAGER BESTAND
```

auszuführen, wobei in diesem Fall auch andere Dateinamen angegeben werden könnten.

## SYSGEN

Normalerweise nichtresidenter Standard-CP/M 2.2-Befehl, für den CPC jedoch gerätespezifisch (Datei SYSGEN.COM)

Beschreibung:

Aufruf: SYSGEN liest das CP/M-Betriebssystem von den Systemspuren einer im Systemformat formatierten Diskette und kopiert es in die reservierten Spuren 0 und 1 einer anderen Diskette. SYSGEN wird zumeist benötigt, um fertig gekaufte Software, die im Vendor-Format geliefert wird, mit dem CP/M-Betriebssystem zu versehen.

```
SYSGEN
```

Es folgt die Aufforderung, zunächst die Quelldiskette mit dem CP/M-Betriebssystem in Laufwerk A zu legen und diese dann gegen die Zieldiskette auszutauschen. Mit

```
SYSGEN *
```

ist nur die Zieldiskette einzulegen, um ein mit MOVCPM geändertes Betriebssystem in die Systemspuren zu übertragen. Soll CP/M aus einer Datei direkt in die Systemspuren geschrieben werden, ist

```
SYSGEN (Dateiname)
```

einzugeben.

Achtung! Die Übertragung des Betriebssystems ist beim CPC nicht ausreichend, um CP/M zu starten. Dies geschieht nämlich mit dem BOOT-Sektor in Spur 0, Sektor 1, der mit BOOTGEN (s.o.) separat auf die Diskette zu schreiben ist.



## TYPE

Residenter CP/M 2.2-Befehl

Beschreibung:

TYPE dient zum einfachen Auflisten von ASCII-Dateien auf dem Bildschirm.

Aufruf: `TYPE (Laufwerk:) Dateiname`

Beispiel: `TYPE PROBE.TXT`

listet die Textdatei PROBE.TXT

## USER

Residenter CP/M 2.2-Befehl

Beschreibung:

USER schaltet zwischen verschiedenen Benutzerbereichen um. Dateien können nämlich nur unter der Benutzernummer aufgerufen werden, in deren Bereich sie abgelegt sind. Die Standardbenutzernummer ist 0.

Aufruf:

```
USER n
```

Dabei gibt n die Nummer des Benutzerbereichs (0 bis 15) an.

Beispiel:

**USER 5**

schaltet in den Benutzerbereich 5 um.



## XSUB

Nichtresidenter CP/M 2.2-Befehl (Datei XSUB.COM)

Beschreibung:

XSUB arbeitet nur im Zusammenhang mit SUBMIT (s.o.) und ermöglicht die bergabe von Parametern auch an aufgerufene Programme oder Dateien, die sonst von Hand eingegeben werden müßten. XSUB ist in einem solchen Fall an erster Stelle in der Befehlsdatei aufzuführen.

Aufruf:

automatisch durch SUBMIT, wenn XSUB in der Befehlsdatei abgelegt ist.

Beispiel:

Nach dem Aufruf des Debuggers DDT soll eine Datei geladen und daraufhin der Speicher zwischen einer vorgegebenen Anfangs- und Endadresse aufgelistet werden. Mit einem normalen SUBMIT-Aufruf können nur Parameter übergeben werden, die sich auf die einzelnen Befehle beziehen (siehe SUBMIT), jedoch nicht auf das Programm DDT. Um dies zu erreichen, muß SUBMIT in der Befehlsdatei enthalten sein.

Zunächst ist die Befehlsdatei folgendermaßen mit einem Texteditor zu schreiben.

```
XSUB
DDT
I$1
R
D$2,$3
```

Hierin steht $l für den Dateinamen der zu bearbeitenden Datei, $2 für die Anfangs- und $3 für die Endadresse. Wenn nun die Befehlsdatei beispielsweise unter dem Namen LIST. SUB gespeichert wird und DDT die Datei WERT.COM in den Speicher laden und zwischen Adresse 100H und 200H auflisten soll. erfolgt der Aufruf mit:

```
SUBMIT LIST WERT.COM 100 200
```

# Anhang

## Steuerzeichen für CP/M-Befehlszeile

| | | |
|---|---|---|
| DEL | CTRL-H | Letzes Zeichen löschen |
| CLR | | Sollte beim CPC nicht verwendet werden! |
| | CTRL-U | Gesamte Zeile löschen, nächste Zeile beginnt mit #-Zeichen |
| | CTRL-X | Gesamte Zeile löschen, Cursor an Zeilenanfang |
| | CTRL-R | Eingabezeile wiederholen |
| | CTRL-E | Fortsetzung in der nächsten Zeile bei überlanger Befehlszeile |
| | CTRL-C | Warmstart auslösen (nur am Zeilenanfang) |
| | CTRL-P | Drucker an- oder abschatten |
| | CTRL-S | Bildschirmausgabe anhalten, zur Fortsetzung beliebige Taste drücken |
| ENTER | CTRL-M | Eingabezeile abschließen |

## ASCII-Code-Tabelle

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| OOH | 0 | NUL | 10H | 16 | 20H | 32 | SP | 30H | 48 | 0 |
| 01H | 1 | SOH | 12H | DLE 17 | 21H | 33 | ! | 31H | 49 | 1 |
| 02H | 2 | STX | 11H | DC1 18 | 22H | 34 | " | 32H | 50 | 2 |
| 03H | 3 | ETX | 13H | DC2 19 | 23H | 35 | # | 33H | 51 | 3 |
| 04H | 4 | EOT | 14H | DC3 20 | 24H | 36 | $ | 34H | 52 | 4 |
| 05H | 5 | ENG) | 15H | DC4 21 | 25H | 37 | % | 35H | 53 | 5 |
| 06H | 6 | ACK | 16H | NAK 22 | 26H | 38 | & | 36H | 54 | 6 |
| 07H | 7 | BEL | 17H | SYN 23 | 27H | 39 | ' | 37H | 55 | 7 |
| 08H | 8 | BS | 18H | ETB 24 | 28H | 40 | ( | 38H | 56 | 8 |
| 09H | 9 | HT | 19H | 25 CAN | 29H | 41 | ) | 39H | 57 | 9 |
| OAH | 10 | LF | 1AH | EM 26 | 2AH | 42 | * | 3AH | 58 | : |
| OBH | 11 | VT | 1BH | SUB 27 | 2BH | 43 | + | 3BH | 59 | ; |
| OCH | 12 | FF | ICH | 28 ESC | 2CH | 44 | , | 3CH | 60 | < |
| ODH | 13 | CR | 1DH | 29 FS | 2DH | 45 | - | 3DH | 61 | = |
| OEH | 14 | SO | 1EH | 30 GS | 2EH | 46 | . | 3EH | 62 | > |
| OFH | 15 | SI | 1FH | 31 RS | 2FH | 47 | / | 3FH | 63 | ? |
| | | | | 31 US | | | | | | |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 40H | 64 | ia | 50H | 80 | P | 60H | 96 | ' | 70H | 112 | p |
| 41H | 65 | A | 51H | 81 | O | 61H | 97 | a | 71H | 113 | q |
| 42H | 66 | B | 52H | 82 | R | 62H | 98 | b | 72H | 114 | r |
| 43H | 67 | C | 53H | 83 | S | 63H | 99 | c | 73H | 115 | s |
| 44H | 68 | D | 54H | 84 | T | 64H | 100 | d | 74H | 116 | t |
| 45H | 69 | E | 55H | 85 | U | 65H | 101 | e | 75H | 117 | u |
| 46H | 70 | F | 56H | 86 | V | 66H | 102 | f | 76H | 118 | v |
| 47H | 71 | G | 57H | 87 | W | 67H | 103 | g | 77H | 119 | w |
| 48H | 72 | H | 58H | 88 | X | 68H | 104 | h | 78H | 120 | x |
| 49H | 73 | I | 59H | 89 | Y | 69H | 105 | i | 79H | 121 | y |
| 4AH | 74 | J | 5AH | 90 | Z | 6AH | 106 | j | 7AH | 122 | z |
| 4BH | 75 | K | 5BH | 91 | [ | 6BH | 107 | k | 7BH | 123 | |
| 4CH | 76 | L | 5CH | 92 | \ | 6CH | 108 | 1 | 7CH | 124 | ı |
| 4DH | 77 | M | 5DH | 93 | ] | 6DH | 109 | m | 7DH | 125 | ) |
| 4EH | 78 | N | 5EH | 94 | ^ | 6EH | 110 | n | 7EH | 126 | - |
| 4FH | 79 | 0 | 5FH | 95 | | 6FH | 111 | o | 7FH | 127 | |



## CTRL-/ASCII-Code

| | | |
|---|---|---|
| CTRL-A | 01H | 1 |
| CTRL-B | 02H | 2 |
| CTRL-C | 03H | 3 |
| CTRL-D | 04H | 4 |
| CTRL-E | 05H | 5 |
| CTRL-F | 06H | 6 |
| CTRL-G | 07H | 7 |
| CTRL-H | 08H | 8 |
| CTRL-I | 09H | 9 |
| CTRL-J | OAH | 10 |
| CTRL-K | OBH | 11 |
| CTRL-L | OCH | 12 |
| CTRL-M | ODH | 13 |
| CTRL-N | OEH | 14 |
| CTRL-0 | OFH | 15 |
| CTRL-P | 10H | 16 |
| CTRL-0 | 11H | 17 |
| CTRL-R | 12H | 18 |
| CTRL-S | 13H | 19 |
| CTRL-T | 14H | 20 |
| CTRL-U | 15H | 21 |
| CTRL-V | 16H | 22 |
| CTRL-W | 17H | 23 |
| CTRL-X | 18H | 24 |
| CTRL-Y | 19H | 25 |
| CTRL-Z | IAH | 26 |

## Stichwortverzeichnis

AMSDOS, 140
Anforderungszeichen, 24
Arbeitsdisketten, 42
Arbeitskopie, 52 ASCII-CODE, 67, 103
ASM, 121, 141
Assembler, 111, 115, 120, 149
Assembler-PRN-Listing, 121
Ausgabekanäle, 86 Backup-Kopie, 72
BASIC, 12 BASIC-Interpreter, 15 BDOS, 24
BDOS-Aufruf, 119
BDOS-Aufrufe, 129
Befehlsdatei, 32, 200
Befehlswort, 118
Befehlszeile, 28
Benutzerbereiche, 64
Betriebssystem, 15
Bezugslaufwerk, 32
Bildschirm, 19
Binäre Werte, 15
BIOS, 24, 26, 134
Bit, 13
Block, 38
Blockbelegungstabelle, 136
Blocking, 137 Blöcke, 57
BOOTGEN, 52, 143
BOOT-Sektor, 52
Byte, 13
CCP, 26
CHKDISK, 43, 144
CLOAD, 145
COPYDISC, 42, 146
CPC6128, 11 CPC-BASIC 17
CPU Z80A, 12 CP/M Plus, 11 CP/M-BOOT-Sektor, 143 CP/M-Start, 23
CSAVE, 147 CTRL-C, 31
CTRL-E, 30 CTRL-H, 30
CTRL-M, 30 CTRL-P, 30
CTRL-Steuerzeichen, 29
CTRL-Taste, 29
CTRL-X, 29
CTRL-Z, 71
Dateiattribute, 63
Dateiausschnitte, 88
Dateien kopieren, **44**
Dateien löschen, 50
Dateien umbenennen, 49
Dateien, 31
Dateiname, 32
Dateinamen, eindeutig, 35
Dateinamen, mehrdeutig, 35
Dateityp, 32
Dateizustand, 57
Datenformat, 41, 174
DDT, 104, 148
Deblocking, 137
Debugger, 148
Debugging, 101
DEL-Taste, 29
Directory, 27, 31, 127
DIR, 32, 157
Disassembler, 111, 152
DISCCHK, 43, **158** DISCCOPY, 42, 159 **DISCKIT2,** 45, 160
Disk-Parameter-Block, 135, 137
Disk-Parameter-Header, 135
Disketten kopieren, 42
Diskettenkapazität, 55
Diskettenlaufwerk, 19
Diskettenparameter, 196
Drucker, 22, 30
DUMP, 102, 161
Editor, 15, 67
ED, 69, 117, 162
Eingabekanäle, 86
Einschaltmeldung, 23
ERA, 50, 172
Erweiterungseinträge, 39
Extension, 32
Extensionen, 34
Extent, 57
Extents, 39, 128
Fehlermeldungen, 51
Festplatte, 22
FILECOPY, **44, 173**
File-Control-Block, 127
Formatierung, 40
FORMAT, 174
FORTH, 17
FORTRAN, 17
Gerätezuordnung, 60
Gerätezuordnungen, 197
Großschrift, 89
Harddisc, 22
Hardware, 24
Hexcode, 104
IBM-Format, 41, 174
Inhaltsverzeichnis, 27

Intel-Hex-Format, 120, 122
ICBYTE, 60
Kleinschrift, 89
Konsole, 19
Label, 118
Laserdrucker, 23
Laufwerkseigenschaften, 58
LOAD, **123,** 175
Maschinencode, 116
Matrixdrucker, 22
Mnemonische Befehle, 115
MOVCPM, 126, 176
M-BASIC, 17
Nichtresidente Befehle, 48
Objektcode, 117
Peripherie, 85 PIP, 79, 177
PRN-Datei, 121
Programmiersprachen, 17
Programmzähler, 113
Prompt, **24**
Prozessorregister, 113
Quelltext, 116 RAM,
12
Record, 38
Records, 57
REN, 187
Residente Befehle, 26, 48
ROM, 12
RS232-Schnittstelle, 138
SAVE, 106, 188
Schreibschutz, 62
Schreibschutz für Dateien, 194
Sektoren, 38
SETUP, 190
Sicherheitskopie, 42, 72
Skewing-Tabelle, 136
Software, **24**
Speicher ändern, 110
Speicher ansehen, 109
Speichererweiterung, 126
Speicherorganisation, 125
Spuren, 38
Stapelverarbeitung, 93
STAT, **35, 55,** 193
SUBMIT, 93, 200
SYSGEN, **52,** 202
Systemdateien, 195
Systemformat, 41, 174
Systemspuren, 41
Tastatur, 19
Text, 67
Texteditor, 117
Textpuffer, 73
Tintenstrahldrucker, 22
TPA, 26
TURBO-PASCAL, 17
Typenraddrucker, **22**
TYPE, 71, 203 USER,
64, 204 V24-
Schnittstelle, 138
Vendor-Format, 41, 174
Verify, 91
Warmstart, 31
WORDSTAR, 19
XSUB, 98, 205
Zeilennummern, 90
Zwischendatei, 73


# Sitzen-Software für Schneider-Computer

## WordStar 3.0 mit MailMerge

Der Bestseller unter den Textverarbeitungsprogrammen für PCs bietet Ihnen bildschirmorientierte Formatierung, deutschen Zeichensatz und DIN-Tastatur sowie integrierte Hilfstexte. Mit MailMerge können Sie Serienbriefe mit persönlicher Anrede an eine beliebige Anzahl von Adressen schreiben und auch die Adreßaufkleber drucken.

**WordStarlMailMerge für den Schneider CPC 464*, CPC 664***
Best.-Nr. MS 101 **(3"-Diskette)**
Best.-Nr. MS 102 (5V," -Diskette im VORTEX-Format)
WordStarlMailMerge für den Schneider CPC **6128**
**Best.-Nr. MS 104 (3"-Diskette)**
**WordStarlMailMerge für den Schneider Joyce PCW 8256**
**Best-Nr. MS 105 (3 "-Diskette)**

Hardware-Anforderungen: Schneider CPC 464*. CPC 664*, CPC 6128 oder Joyce, beliebiger Drucker mit Centronics-Schnittstelle
Der Standard-Speicherplatz beim CPC 464'664 erlaubt ohne Speichererweiterung Blockverschiebe-Operationen nur bedingt und Simultan-Drucken gar nicht.

**Dieses Programm kostet**

**DM 199,-** inkl. MwSt. Unverbindliche Preisempfehlung

Ii^v1 '
Markt&Technik
**Schneider CPC Software**

**(1I.**

**Wör"dStar 30**

**mit MailMerge für den Schneider**
CPC 464/664
3" schnvider-Formar

## Und dazu die weiterführende Literatur:

Mit diesem Buch haben Sie eine wertvolle Ergänzung zum WordStar-Handbuch: Anhand vieler Beispiele steigen Sie mühelos in die Praxis der Textverarbeitung mit Wordstar ein. Angefangen beim einfachen Brief bis hin zur umfangreichen Manuskripterstellung zeigt Ihnen dieses Buch auch, wie Sie mit Hilfe von MailMerge Serienbriefe an eine beliebige Anzahl von Adressen mit persönlicher Anrede senden können.
Best.-Nr. MT 779
**ISBN 3-89090-180-8** **DM 49,-**

**Erhältlich bei Ihrem Buchhändler.**

Markt &Technik-Produkte erhalten Sie in den Fachabteilungen der Warenhäuser, im Versandhandel, in Computerfachgeschäften oder bei Ihrem Buchhändler.

Markt&Technik
Zeltschriften - Bück' r
Software , Schulung

Mark!&Techn Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haarbei München, Telefon (089 4613-0

# Sitzen-Software für Schneider-Computer

## dBASE II, Version 2.41

dBASE II. das meistverkaufte Programm unter den Datenbanksystemen, eröffnet Ihnen optimale Möglichkeiten der Daten- und Dateihandhabung. Einfach und schnell können Datenstrukturen definiert, benutzt und geändert werden. Der Datenzugriff erfolgt sequentiell oder nach frei wählbaren Kriterien, die integrierte Kommandosprache ermöglicht den Aufbau kompletter Anwendungen wie Finanzbuchhaltung, Lagerverwaltung. Betriebsabrechnung usw.

**dBASE II für den Schneider CPC 464*, CPC 664***
**Best.-Nr. MS 301 (3 "-Diskette)**
**Best.-Nr. MS 302 (5S,"-Diskette im VORTEX-Format)**
**dBASE** 11 für den Schneider CPC **6128**
**Best.-Nr. MS 304 (3 "-Diskette)**
**dBASE II für den Schneider Joyce PCW 8256**
**Best.-Nr. MS 305 (3 "-Diskette)**

Hardware-Anforderungen: Schneider CPC 464`, CPC 664*. CPC 6128 oder Joyce, beliebiger Drucker mit Centronics-Schnittstelle * dBASE II für den Schneider CPC 464664 ist lauffähig mit einer Speichererweiterung auf 128 Kbyte.

s w
Markt&Technik
**Schneider CPC Software**

**dBASE** II

ITrry TT!h
**für den Schneider CPC 6128**
3" Schneider-Format

**Dieses Programm kostet**

**DM 199,-** **inkl. MwSt.** Unverbindliche Preisempfehlung

## Und dazu die weiterführende Literatur:

Zu einem Weltbestseller unter den Datenbanksystemen gehört auch ein klassisches Einführungs- und Nachschlagewerk! Dieses Buch des deutschen Erfolgsautors Dr. Peter Albrecht begleitet Sie mit nützlichen Hinweisen, die nur von einem Profi stammen können, bei Ihrer täglichen Arbeit mit **dBASE** II. Schon nach Beherrschung weniger Befehle ist der Einsteiger in der Lage, Dateien zu erstellen, mit Informationen zu laden und auszuwerten
Best.-Nr.
**3-89090-188-3** **DM 49,-**

Erhältlich bei Ihrem Buchhändler.

Markt &Technik-Produkte erhalten Sie in den Fachabteilungen der Warenhäuser, im Versandhandel, in Computerfachgeschäften oder bei Ihrem Buchhändler.

tt - Akt&
Markt&Technik
3r~r-n-Büc1er
Schulung

2,.3013Hoaro, iVi meh·e TSe(o 89!14613-0

# Spitzen-Software für Schneider-Computer

## Multiplan, Version 1.06

Wenn Sie die zeitraubende manuelle Verwaltung tabellarischer Aufstellungen mit Bleistift, Radiergummi und Rechenmaschine satt haben, dann ist MULTIPLAN, das System zur Bearbeitung «elektronischer Datenblätter«, genau das richtige für Siel Das benutzerfreundliche und leistungsfähige Tabellenkalkulationsprogramm kann bei allen Analyse- und Planungsberechnungen eingesetzt werden wie z. B. Budgetplanungen. Produktkalkulationen, Personalkosten usw. Spezielle Formatierungs-, Aufbereitungs- und Druckanweisungen ermöglichen außerdem optimal aufbereitete Präsentationsunterlagen

**MULTIPLAN für den Schneider CPC 464*, CPC 664***
**Best.-Nr. MS 201 (3 "-Diskette)**
**Best.-Nr. MS 202 (5' / "-Diskette im VORTEX-Format)**
**MULTIPLAN für den Schneider CPC 6128**
**Best.-Nr. MS 204 (3 "-Diskette)**
**MULTIPLAN für den Schneider Joyce PCW 8256**
**Best.-Nr. MS 205 (3"-Diskette)**

Hardware-Anforderungen: Schneider CPC 464*, CPC 664`. CPC 6128 oder Joyce, beliebiger Drucker mit Centronics-Schnittstelle * MULTIPLAN für den SchneiderCPC 464:664 ist lauffähig mit einer Speichererweiterung auf 128 Kbyte.

h1a,1r 11nik
**Schneider C M -Software**

**MICROSOFT MULTIPLAN**
**für den Schneider CPC 464**
3" Schneider-Format

**Dieses Programm kostet**

**DM 199,-** inkl. MwSt. Unverbindliche Preisempfehlung

## Und dazu die weiterführende Literatur:

Dank seiner Menütechnik ist **MULTIPLAN** sehr schnell erlernbar. Mit diesem Buch von Dr. Peter Albrecht werden Sie Ihre Tabellenkalkulation ohne Probleme in den Griff bekommen. Als Nachschlagewerk leistet es auch dem Profi nützliche Dienste.

Best: Nr. MT **835**
**ISBN 3-89090-186-7** **DM 49,-**

Erhältlich bei Ihrem **Buchhändler.**

Markt&Technik-Produkte erhalten Sie in den Fachabteilungen der Warenhäuser. im Versandhandel. in Computerfachgeschäften oder bei Ihrem Buchhändler.

m
ö

Markt&Technik

Zei- hriften . Becher
Software . Schu·ung

Markt&Techn k ✓erhö AG, Buchverlag, Hons P Esel-Str 8013 Maier oei P. Ln,hee. Telefon (089) 4613-0

# JETZT AUF SCHNEIDER-COMPUTERN:

## DIE PROGRAMM-BIBLIOTHEK FÜR
# TURBO PASCAL®

### TURBO-Lader-Grundpaket

Das TURBO-Lader-Grundmodul ist eine umfangreiche Programm-Bibliothek für den TURBO-Pascal-Programmierer. Sie umfaßt zahlreiche ausführlich dokumentierte Prozeduren und Funktionen, dieder Profi zur schnellen Lösung seiner Programmieraufgaben verwenden kann.

Das **TURBO-Lader-Grundpaket** erfordert den **TURBO-Pascal-Compi**ler. Es ist lieferbar auf **3"- und 5'la"-Disketten und lauffähig auf dem Schneider CPC 464, CPC 664, CPC 6128.**

3"-Disk. Best.-Nr. MS 413 5'.
<"-Disk. Best-Nr. MS 415

**DM 138,-* *** inklusive MwSt.

### TURBO-Lader Business

TURBO-Lader Business umfaßt einen komfortablen Bildschirm-Maskengenerator und eine professionelle Dateiverwaltung. Der Maskengenerator gibt dem Pascal-Programmierer ein Werkzeug zur einfachen Bearbeitung von Bildschirm-Masken in die Hand.

**TURBO-Lader Business erfordert den TURBO-Pascal-Compiler und das TURBO-Lader-Grundpaket. Es ist lieferbar auf 3"- und 5'1/4"-Disketten und lauffähig auf dem Schneider** CPC 464, CPC **664,** CPC **6128.**

3"-Disk. Best.-Nr. MS 423 5',"-
Disk. Best.-Nr. MS 425

**DM 148,* '** inklusive MwSt.

### TURBO-Lader Science

TURBO-Lader Science ist eine Sammlung technisch/wissenschaftlicher Funktionen und professioneller statistischer Verfahren für die Bereiche Medizin, Betriebs- und Volkswirtschaft, Technik und Naturwissenschaften.

**TURBO-Lader Science erfordert den TURBO-Pascal-Compiler und das TURBO-Lader-Grundpaket. Es ist lieferbar auf 3"- und 5'1,"-Disketten und lauffähig auf dem Schneider** CPC 464, CPC **664,** CPC **6128.**

3"-Disk. Best.-Nr. MS 433 5' ,
"-Disk. Best.-Nr. MS 435

**DM 189,*** inklusive MwSt.

Übrigens können Sie auch folgende Turbo-Pascal-Produkte für Schneider CPC und Joyce bei Markt&Technik beziehen:
Turbo Pascal 3.0. Turbo Pascal 3.0 mit Grafikunterstützung,
Turbo Tutor (deutsch), Turbo Tutor (englisch), Turbo Graphix Toolbox, Turbo Toolbox.

TURBO Pascal ist ein Warenzeichen der Borland Inc.. USA. TURBO-Lader. TURBO-Lader Business und TURBO-Lader Science sind Warenzeichen der Fa. Lauer & Wallnitz.

Markt&Technik-Produkte erhalten Sie in den Fachabteilungen der Warenhäuser, im Versandhandel. in Computerfachgeschäften oder bei Ihrem Buchhändler.

Markt&Technik

Zeitschriften - Büch"
Software - Schulung

Mark-&Techn kVe'lagAG, Buchverag, Hans-Pinsel-SfraSe 2, 8013 Haarbei München, Telefon (089) 4613-0

JÜRGEN HÜCKSTÄDT
Jahrgang 1949, ist Diplomingenieur für Wasserbau. Im Rahmen seiner langjährigen Tätigkeit in einem Ingenieurbüro kam er schon frühzeitig mit Personal Computern in Berührung. Er entwickelte verschiedene Softwarepakete für wasserwirtschaftliche und geodätische Anwendungen und wurde schließlich mit der Leitung der EDV-Abteilung beauftragt. Seit einigen Jahren leitet er Seminare für BASIC- und Assemblerprogrammierung und ist darüber hinaus auch als Übersetzer, Lektor und Autor von EDV-Fachliteratur tätig. Bereits erschienene Bücher in unserem Verlag vom selben Autor:

- Basic 7.0 auf dem Commodore 128 (Best.-Nr. 90149)
- Der Schneider CPC 6128 ( Best.-Nr. 90192)

# CP/M 2.2 Anwender-Handbuch CPC 464/664/6128

Die Schneider CPC-Computer erfreuen sich immer größerer Beliebtheit, nicht zuletzt deshalb, weil sie die Betriebsart CP/M zulassen. Jetzt können Sie mit Ihrem Schneider-Computer auf die größte Softwarebibliothek der Welt zurückgreifen, wodurch Ihnen ungeahnte Möglichkeiten offenstehen.
Dieses Buch ist sowohl für den Anfänger als auch für den Profi ein wichtiges Arbeitsbuch und Nachschlagewerk. Es behandelt CP/M 2.2 nicht nur in seiner allgemeinen Form, wie sie für sämtliche CP/M-Computer gültig ist, sondern bezieht auch die Hardware der CPC-Computer mit ein. Sie finden hier alle systemspezifischen Informationen über Floppy, Tastatur und Bildschirm, die für das Arbeiten mit CP/M auf den Schneider-Computern von Bedeutung sind.
Auf den einfachsten Grundlagen aufbauend erfahren Sie alles über dieses universelle Betriebssystem, das gleichermaßen auf dem CPC 464, 664 und 6128 lauffähig ist. Neben den theoretischen Grundlagen sind viele praktische Beispiele enthalten um das Lernen zu erleichtern.

Aus dem Inhalt:

- einfache Grundlagen
- ausführliche Erläuterung sämtlicher CP/M 2.2-Befehle
- alles über Dateien
- Assemblerprogrammierung und Debugging
- interne Speicherorganisation und Sprungtabellen
- Anpassung von CP/M-Software an CPC-Computer

ISBN 3-89090-204-9

DM 46,-